UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UM SISTEMA DE RECONHECIMENTO DE PEÇAS BASEADO EM REDES NEURAIS

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA

IDMILSON HABER SEPEDA FILHO

FLORIANÓPOLIS, AGOSTO DE 1994

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UM SISTEMA DE RECONHECIMENTO DE PEÇAS BASEADO EM REDES NEURAIS

IDMILSON HABER SEPEDA FILHO

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE

MESTRE EM ENGENHARIA

ESPECIALIDADE ENGENHARIA ELÉTRICA, ÁREA DE CONCENTRAÇÃO: CONTROLE, AUTOMAÇÃO E INFORMÁTICA INDUSTRIAL, E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO.

PROF. MARCELO RICARDO STEMMER, Dr. -Ing.
Orientador

PROF. ROBERTO DE SOUZA SALGADO. Ph.D.
Coordenador da Pós-Graduação Eng. Elétrica

Banca Examinadora:

PROF. MARCELO RICARDO STEMMER, Dr. -Ing. (Presidente)

PROF. FERNANDO MENDES DE AZEVEDO, Ph.D.

PROF. GUILHERME BITTENCOURT, Dr.

PROF. RENATO GARCIA OJEDA, Sc.D.

a Deus e aos meus pais

AGRADECIMENTOS

Gostaria de agradecer especialmente ao Prof. Marcelo Ricardo Stemmer por sua orientação e amizade no decorrer deste trabalho.

Agradeço ao PICD-UFPa/CAPES pelo apoio financeiro.

Agradeço também ao Prof. Armando A. Gonçalves Jr., André Manzoli e Danilo do Labmetro/CERTI, pelo apoio material.

Agradeço a Ana Luiza, Celito e família, e Letícia, Ronchi e família por toda atenção dispensada.

Agradeço a meus amigos de república Alexandre, Aberides, Luis Otavio e Junior e a todos os amigos com quem convivi durante o trabalho.

Agradeço a Cristiane pelo carinho dispensado em todos os momentos.

Agradeço principalmente a Idmilson (pai), Sebastiana, Ideuzana, Ideuzanira, Ivana e Ivanilson.

SUMÁRIO

RESUMO

O reconhecimento de objetos por computador é uma área de grande interesse. Novos paradigmas como redes neurais estão sendo utilizados com o propósito de conseguir sistemas que sejam robustos e que executem esta tarefa rapidamente. Neste trabalho foram estudadas várias arquiteturas e métodos de treinamento de redes neurais. Foi escolhida uma arquitetura (counterpropagation) baseando-se nela foi implementado um pacote de software para reconhecimento de peças mecânicas. Foram utilizados dois tipos de algoritmo de pré-processamento: um utilizando curvas de distribuição de intensidade e outro usando o histograma de níveis de cinza. Os resultados obtidos até o momento demonstram a adequação da técnica de redes neurais para esta aplicação. Finalizando, são feitas algumas observações resultantes da experiência adquirida com o trabalho e algumas sugestões são feitas.

ABSTRACT

The object recognition by computer is an area of great interest. New paradigms as neural networks are now being used to obtain system that are robust and solve this task faster. In this work several architectures and training algorithms of neural networks were studied. An architecture (Counterpropagation) has been chosen and from this a software package was implemented for mechanical workpiece recognition. Two kinds of algorithms for image processing have been used: one of them used intensity distribution curves and the other used a gray level histogram. The obtained results are satisfactory and demonstrate the adequacy of the neural networks technical for workpiece recognition. Finally, some observations from experience with this work are made and some suggestions are given.

CAPÍTULO I

INTRODUÇÃO GERAL

O tema de redes neurais tornou-se um importante campo de pesquisa a partir da segunda metade da década de 80, apesar de que as primeiras contribuições na área tenham surgido já na década de 40. Nesta época, foram realizadas importantes pesquisas sobre a natureza da atividade neural do cérebro humano, que resultaram em novos conceitos e teorias a respeito da estrutura e funcionamento biológico das redes neurais. Com o progresso em neuroanatomia e neurofisiologia, fisiologistas desenvolveram modelos de aprendizado humano. Um desses modelos foi o de D. O. Hebb [1], que em 1949 propôs uma lei de aprendizado que veio a ser o ponto de partida para vários algoritmos de treinamento de redes neurais.

A partir destes trabalhos, tem-se desde então procurado desenvolver esquemas e técnicas capazes de realizar funções até então atribuídas exclusivamente ao cérebro humano, tais como a capacidade de reconhecer padrões, memorizar e aprender pela experiência, entre outros.

Na década de 60, a Inteligência Artificial apresentava duas tendências bem definidas e concorrentes. A primeira, baseada no processamento simbólico, constitui a base da chamada inteligência artificial tradicional, na qual o sistema de computação é visto como uma mecanização seqüencial de processos ou eventos. A segunda tendência, baseada no processamento paralelo distribuído e nas redes neurais, surgiu como tentativa de imitar as estruturas neurais biológicas. Nesta tendência, a informação contida no sistema é representada por padrões de ativação de elementos semelhantes ao neurônio biológico. Cada elemento executa uma função básica simples independente de outros elementos e em paralelo com eles. Os elementos são interligados em forma de rede conforme uma determinada topologia. Ambas as correntes pareciam promissoras mas, devido a uma série de fatores, as técnicas tradicionais receberam inicialmente maior interesse. Entre estes fatores pode-se citar [1,2]:

- sucesso prático na implementação da arquitetura computacional Von Neumann;
- rápidos avanços de Hardware e Software de apoio ao processamento seqüencial;
- limitada capacidade dos sistemas computacionais paralelos da época;
- limitado conhecimento de diversas características biológicas fundamentais do sistema neural, que resultaram em dificuldades de implementação das técnicas e em resultados práticos ainda pouco significativos;
- conclusões parciais negativas a respeito do potencial das redes neurais, baseadas nas topologias e métodos de aprendizado em estudo na época.

Por outro lado, em consequência da aplicação das técnicas seqüenciais convencionais a inúmeras áreas, muitas dificuldades inerentes ao processamento seqüencial tornaram-se evidentes. Aplicações industriais relacionadas, entre outros, ao processamento de conhecimentos, robótica [3], classificação de padrões [4], encontraram problemas que sobrecarregavam o processamento simbólico dos programas da corrente tradicional. Este fato, associado ao surgimento de novas teorias biológicas, novos modelos neurais e novos algoritmos de treinamento, produziu uma retomada do interesse pelas redes neurais em anos recentes.

Várias características das redes neurais tem-se mostrado extremamente promissoras na procura de soluções para problemas nos quais as técnicas tradicionais não levaram a resultados aceitáveis. Entre eles, pode-se citar o paralelismo, que permite analisar vários parâmetros simultaneamente e reduzir o tempo computacional, a capacidade de aprender através de exemplos, imunidade considerável à ruídos no sinal de entrada etc.

As vantagens citadas conduziram à aplicação crescente da técnica de redes neurais às mais diversas áreas, tais como:

- Projeto de filtros adaptativos;
- Modelagem, otimização e controle de processos [11];
- Reconhecimento de voz e escrita;
- Visão computacional, para aplicação em robótica [3];
- Processamento de sinais;
- Previsão de carga elétrica;
- Diagnóstico de falhas.

Existem muitas demonstrações das capacidades de redes neurais artificiais: uma rede foi treinada para converter texto em representações fonéticas, a qual então as convertia para voz por outro método; outra rede podia reconhecer caracteres escritos a mão; e um sistema baseado em redes neurais foi utilizado para compressão de imagem [1].

Grande variedade de novas aplicações vem sendo estudadas e desenvolvidas em centros de pesquisa, universidades e grandes empresas em todo o mundo [3,4,5,6,7,8].

Dentre estas, a visão computacional é uma área de grande interesse com várias aplicações em automação industrial, em especial na área de robótica. Novos paradigma, como redes neurais, estão sendo utilizados com o propósito de conseguir sistemas que sejam robustos e que executem esta tarefa rapidamente.

Os computadores digitais são capazes de efetuar milhares de cálculos com pontos flutuantes por segundo, mas são bastantes lentos no reconhecimento de formas em uma imagem visual, devido à inadequação dos algoritmos disponíveis e à própria estrutura de processamento da informação. A tarefa de reconhecer um objeto em uma imagem é tão habilmente feita por um humano, embora os neurônios tenham o tempo de chaveamento maior que os circuitos eletrônicos de hoje [1], devido à arquitetura de inter-conexão dos neurônios no cérebro humano.

Ela difere bastante da estrutura dos computadores com arquitetura Von Neumann, que foram projetados para realizarem tarefas seqüenciais, que devem ser descritas passo a passo por um programador. Considere-se, por exemplo, o quanto seria complexo reconhecer através de um programa seqüencial um cão em uma imagem. Seria necessário descrever todas as possíveis posições nas quais poderíamos encontra-lo, todos os diferentes tamanhos de cada cão e ainda todas as diferentes raças existentes e outros dados como cor, pelagem etc. Em muitas aplicações do mundo real, deseja-se que os computadores realizem o reconhecimento de padrões complexos, tal como descrito anteriormente. Entretanto, computadores seqüenciais são claramente inadequados para solução deste tipo de problema. Para solucionar esta questão buscou-se aspectos da fisiologia do cérebro humano como base para o novo modelo de processamento, daí o nome de redes neurais artificiais ou simplesmente redes neurais[1].

As redes neurais são compostas de elementos que realizam muitas funções análogas as funções elementares do neurônio biológico. Além da semelhança superficial com a anatomia do cérebro, essas redes exibem algumas características do cérebro humano. Por exemplo, elas aprendem por experiência. Mas apesar dessas similaridades funcionais, redes neurais artificiais estão longe de duplicar as funções do cérebro humano.

Redes neurais artificiais podem modificar seu comportamento em resposta a seu ambiente, ou seja, mostrando-se um conjunto de entradas (talvez com saídas desejadas), as redes ajustam os seus pesos para produzir um conjunto de saídas consistente. Este fato, mais que qualquer outro, é responsável pelo interesse que vêm recebendo. Diz-se então que ela pode aprender (***learn***). Existe uma grande variedade de algoritmos de treinamento, todos com seus pontos fortes e fracos, que serão analisados no capítulo 2.

Uma vez treinada, uma resposta da rede pode ser insensível a pequenas variações na sua entrada. Esta habilidade é essencial para o reconhecimento de padrões no mundo real, por

---

[1] Existem duas traduções para o termo ***neural*** proveniente da lingua inglesa. Alguns autores traduzem como "neuronal" e outros como "neural". Neste trabalho, se usará o termo "neural", por razões de brevidade.

causa de ruídos ou distorções de padrões (imperfeições) do mundo em que vivemos. É importante notar que os resultados das redes são obtidos a partir de sua estrutura e não pelo uso de inteligência humana embutida em alguma forma de programa de computador. Algumas redes neurais são capazes de abstrair a essência de um conjunto de entradas. Por exemplo, uma rede pode ser treinada por versões destorcidas da letra A, e após um treinamento adequado, produzir uma letra A perfeita, deste modo aprendendo algo que ela nunca viu antes.

Apesar de todas essas vantagens, redes neurais são claramente impróprias para tarefas como cálculo de uma folha de pagamento, sendo mais aplicadas a uma classe de tarefas de reconhecimento de padrões.

Como as redes neurais imitam a estrutura do cérebro humano, elas detêm um grau de não previsibilidade. A menos que todas as entradas possíveis tenham sido testadas, não existe maneira para se ter certeza se a saída é precisa. Poder-se-ia dizer que qualquer erro seria intolerável em determinadas situações (em um sistema de defesa do espaço aéreo, por exemplo). É necessário considerar, no entanto, que o ser humano pode também cometer enganos sob as mesmas circunstâncias.

O objetivo deste trabalho foi desenvolver um sistema que permitisse a identificação de peças mecânicas sendo transportadas em uma esteira e que devem ser seletivamente manipuladas por um robô para posterior processamento em uma célula flexível de manufatura.

A imagem da peça é obtida por uma câmera CCD (Charge Coupled Device) e digitalizada. A imagem digitalizada é processada de forma a permitir uma identificação da peça após uma etapa inicial de aprendizado da rede neural. Além disso, o sistema desenvolvido é capaz de fornecer a posição (dando a translação da peça em relação a sua localização na imagem) e orientação da peça na esteira (informando a sua rotação). A orientação é informada em termo de cosseno e seno do ângulo que a peça está rotacionada.

O resultado obtido na saída da rede neural será passado para o Gerente da célula flexível de manufatura, que através de uma transformação de coordenadas informa ao robô a localização da peça.

A estrutura da rede e o algoritmo de aprendizado mais adequados para esta tarefa foram escolhidos e implementados. A escolha do algoritmo de aprendizado foi feita após uma análise de vários modelos neurais; algumas tentativas com outros modelos de redes não apresentaram resultados satisfatórios. Posteriormente, foram realizados testes operacionais do sistema com peças reais de uma determinada família. Optou-se, nesta primeira versão, por peças rotativas simétricas, que permitem uma identificação a partir de uma imagem bidimensional.

No capítulo 2 serão descritas as arquiteturas mais conhecidas e os métodos de aprendizado mais utilizados em redes neurais.

No capítulo 3, serão apresentados alguns métodos básicos de processamento de imagem, utilizados aqui para a preparação dos dados visuais para alimentar a rede neural.

A seguir, no capítulo 4, será discutido, analisado e mostrada a implementação um sistema de reconhecimento de peças baseado em redes neurais.

Finalmente, as conclusões e perspectivas de futuras pesquisas serão apresentadas no capítulo 5.

CAPÍTULO II

ARQUITETURA E MÉTODOS DE APRENDIZADO NEURAL

2.1 - Conceitos Fundamentais de Redes Neurais Artificiais

Embora ainda se conheça pouco sobre a operação completa do cérebro humano, as redes neurais são baseadas nesta estrutura.

O sistema nervoso humano, constituído de células chamadas neurônios, é extremamente complexo. São estimados 10^{11} neurônios participando em aproximadamente 10^{15} interconexões sobre vias de transmissão que podem variar de alguns milímetros a um ou mais metros. Cada neurônio compartilha muitas características com outras células do corpo, mas uma capacidade única para receber, processar, e transmitir sinais eletroquímicos sobre as vias neurais do sistema de comunicação do cérebro.

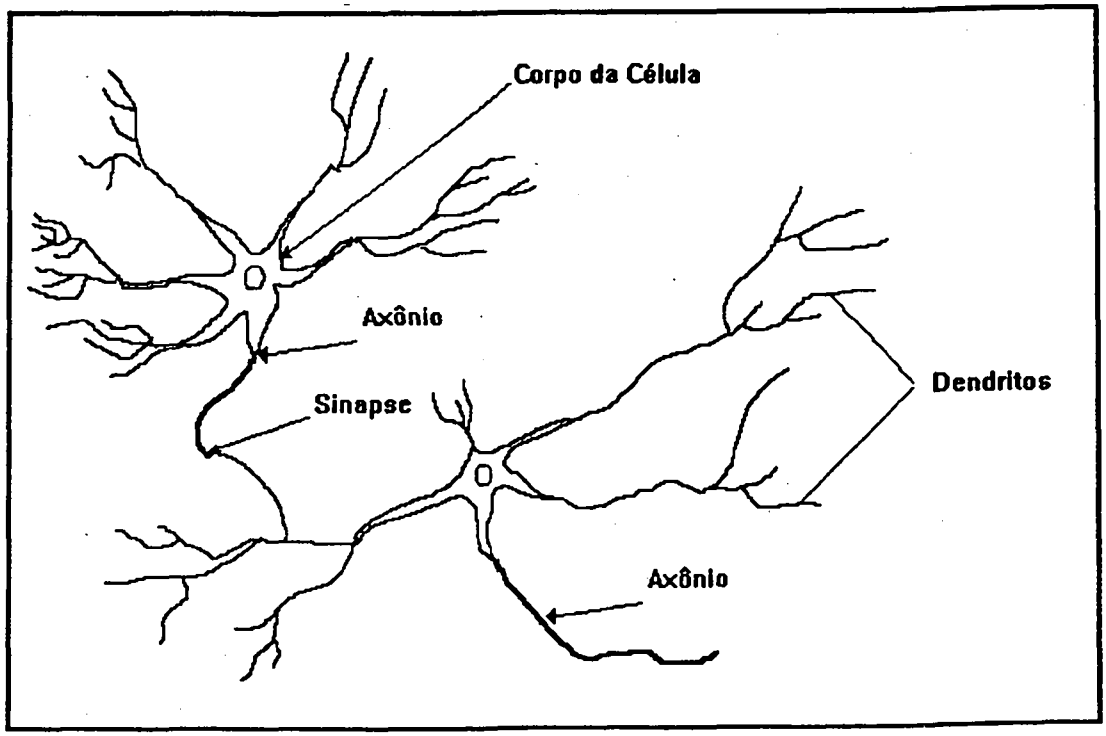

FIGURA 2.1 - Neurônio biológico

A figura 2.1 mostra um neurônio biológico. O neurônio tem um corpo e várias ramificações. **Dendritos** conduzem sinais das extremidades para o corpo da célula [29], onde são somados. O ponto de conexão de dois neurônios é chamado **sinapse**. Algumas entradas tendem a excitar a célula, outras tendem a inibir seu disparo. Quando a excitação acumulada na célula excede um limite, a célula dispara, enviando um sinal através do **axônio** para outros neurônios.

O neurônio artificial foi projetado para imitar as características de 1ª ordem do neurônio biológico. De fato, um conjunto de entradas é aplicado, cada uma produzindo uma saída para outro neurônio. Cada entrada é multiplicada por um peso correspondendo analogamente à força sináptica, e todas as entradas ponderadas são então somadas para determinar o nível de ativação do neurônio. A figura 2.2 mostra o modelo que implementa esta idéia.

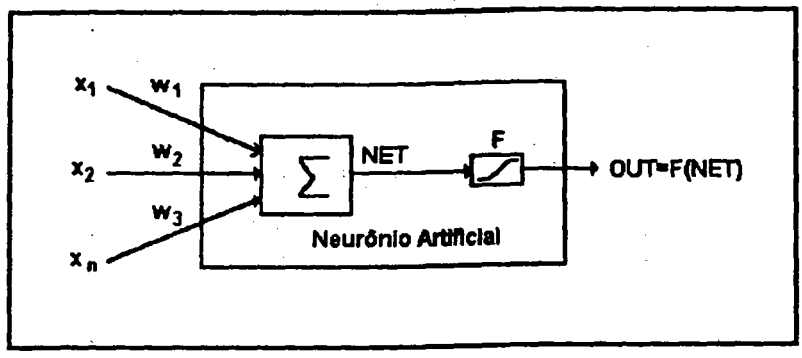

FIGURA 2.2 - Neurônio artificial com função de ativação

Para formalizar, pode-se representar isso numa notação vetorial como segue:

$$\mathbf{NET} = \mathbf{XW} \tag{2.01}$$

onde:

NET = soma algébrica das entradas ponderadas

X = vetor de entrada

W = vetor de pesos

O resultado da soma (**NET**) é processado por uma função de ativação (**F**) para produzir o sinal de saída do neurônio (**OUT**). Para tal, uma função (F) limitada deve ser

escolhida. A função de ativação pode ser entendida como se fosse um ganho não linear para o neurônio artificial. Esse ganho possibilita manusear tanto sinais pequenos quanto sinais grandes, solucionando o dilema de saturação-ruído. A função sigmóide é frequentemente escolhida por ser um função logística diferenciável e monotônica. Esta função é expressa matematicamente como :

$$F(x) = 1/(1+e^{-x}) \tag{2.02}$$

Portanto,

$$OUT = 1/(1+e^{-NET}) \tag{2.03}$$

A região central da função logística soluciona o problema de processar pequenos sinais, enquanto suas regiões de ganho decrescente nas extremidades positiva e negativa são apropriadas para grandes excitações.

É conveniente considerar os pesos como uma matriz W, com dimensões n x m, onde n é o número de entradas e m o número de neurônios.

Uma rede neural multicamadas é obtida pela associação de diversos neurônios como visto anteriormente. Redes neurais artificiais multicamadas são maiores, mais complexas e oferecem maior capacidade computacional que um neurônio isolado.

Redes multicamadas podem ser formadas simplesmente colocando em cascata redes de 1 camada; a saída de uma camada é utilizada como entrada para a camada subseqüente. A figura 2.3 mostra tal rede. Nesta figura o neurônio artificial é representado apenas por N, entretanto ele segue o modelo daquele neurônio mostrado na figura 2.2.

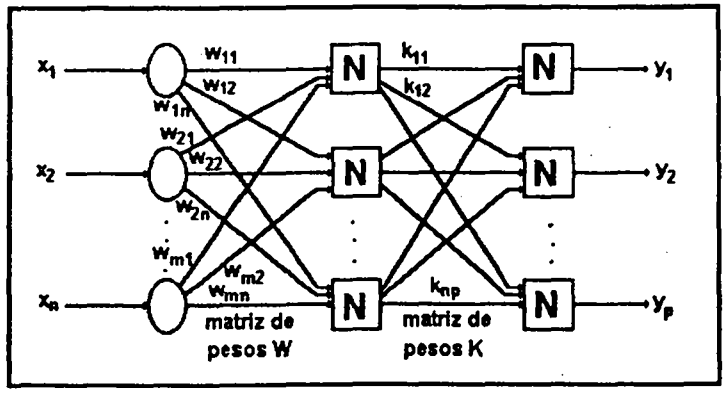

FIGURA 2.3 - Rede neural de duas camadas

Redes neurais artificiais e biológicas podem ter muitas das conexões desfeitas, mas a conectividade total é mostrada por razões de generalidade.

A função de ativação usada em redes multicamadas deve ser não linear, para que haja um aumento do poder computacional. O cálculo da saída de uma camada consiste da multiplicação do vetor de entrada pela primeira matriz de pesos, e então (se não existe função de ativação não linear) multiplica o vetor resultante pela segunda matriz de pesos. Isto pode ser expresso como:

$$(XW1)W2 \tag{2.04}$$

Desde de que a multiplicação é associativa, tem-se:

$$X(W1W2) = XW \tag{2.05}$$

Isto mostra que uma rede linear de duas camadas é exatamente equivalente a uma de uma única camada.

As redes neurais podem ser recorrentes ou não recorrentes dependendo se possuem ou não realimentação, respectivamente. As redes recorrentes exibem propriedades de memória, porque suas saídas atuais dependem das saídas anteriores.

De todas as características das redes neurais artificiais, nenhuma captura a imaginação, como sua habilidade para aprender. Seu treinamento mostra muitos paralelos com o desenvolvimento intelectual humano, mostrando que pode ser que tenhamos alcançado um entendimento fundamental deste processo.

Uma rede é treinada tal que a aplicação de um conjunto de entradas produz o desejável (ou no mínimo consistente) conjunto de saídas. O treinamento é realizado pela aplicação sequencial de vetores de entrada, enquanto o ajuste dos pesos da rede é efetuado por um procedimento predeterminado. Durante o treinamento os valores dos pesos convergem para valores que produzem a saída desejada.

O treinamento pode ser supervisionado e não supervisionado. O treinamento supervisionado exige um vetor de entrada com um vetor alvo representando a saída desejada,

juntos são chamados o par de treinamento. No treinamento não supervisionado o conjunto de treinamento consiste somente de vetores de entrada. O algoritmo de treinamento modifica os pesos da rede para produzir o vetor de saída que é consistente, isto é, a aplicação de um vetor de treinamento na entrada ou um vetor que é suficientemente similar à ele irá produzir o mesmo padrão de saída.

Existem muitos modelos de redes neurais artificiais A tabela 2.1 mostra os principais modelos [8, 30].

Modelo Neural	Principais Aplicações	Pontos Fortes	Pontos Fracos
Perceptrons	Reconhecimento de caracteres.	Rede neural mais antiga.	Não reconhece padrões complexos; sensível à mudanças.
Backpropagation	Larga aplicação	Rede mais popular, trabalha bem, e é simples para aprender	Treinamento supervisionado com exemplos abundantes
Counterpropagation	Reconhecimento de padrões, análise estatística etc.	Menos geral quando comparada a Back-propagation	Grande número de EP[1]s e conexões
Hopfield	Recuperação de dados e fragmentos de imagens.	Implementação em larga escala.	Sem aprendizado, os pesos devem ser inicializados.
Memória Associativa Bidirecional	Reconhecimento de padrões.	Estável	Precisa muitos EP's para armazenar poucos padrões

Tabela 2.1 - Modelos de redes neurais mais conhecidos

Uma análise mais detalhada de cada um deles será feita a seguir.

2.2 - PERCEPTRONS

2.2.1 - Introdução

McCulloch e Pitts (apud [31]) publicaram o primeiro estudo sistemático de redes neurais artificiais. Em trabalhos posteriores exploraram os paradigmas para reconhecimento de padrões a despeito da translação e rotação. Seus trabalhos envolviam um simples neurônio mostrado na figura 2.2.

[1]EP - elemento de processamento

Perceptrons são redes neurais que consistem de uma única camada de neurônios artificiais conectados por pesos a um conjunto de entradas.

Nos anos 60, a rede perceptrons criou um grande campo de interesse e otimismo e foram feitas demonstrações convincentes da potencialidade dessas redes. A euforia inicial foi trocada por desilusão quando descobriu-se que elas falhavam em certas tarefas de simples aprendizado. Minsky (apud [1]) analisou este problema com grande rigor e provou que existem severas restrições no que uma rede perceptron de uma camada pode representar, e daí no que ela pode aprender.

Apesar das limitações das perceptrons, elas têm sido extensivamente estudadas (se não largamente utilizadas). Sua teoria é a base para muitas outras formas de redes neurais artificiais e demonstram princípios importantes. Por essas razões, perceptron é um ponto de partida para o estudo de redes neurais artificiais.

2.2.2 - Representação de Perceptrons

A prova do teorema de aprendizado perceptron demonstra que uma rede perceptron poderia aprender qualquer coisa que ela pudesse representar [1]. É importante distinguir entre representação e aprendizado. Representação se refere a habilidade de uma rede perceptron (ou outra rede) simular uma função específica. O aprendizado exige a existência de um procedimento sistemático para ajustar os pesos da rede para produzir aquela função.

- Problema do ou-exclusivo:

Um dos resultados de Minsky, mostra que uma rede perceptron de uma camada não pode simular uma simples função ou-exclusivo. Esta função aceita duas entradas que podem ser **0** ou **1**. Ela produz uma saída **1** somente se uma entrada é **1** (mas não ambas).

Uma rede perceptron de uma camada não pode ser treinada para resolver problemas relacionadas a funções não linearmente separáveis, como no caso do ou-exclusivo.

2.2.3 - Eficiência da armazenagem

Existem sérias questões sobre a eficiência na armazenagem de perceptron (e de outras redes neurais) relativas à memória convencional do computador e métodos para recupera-la. Por exemplo, poderia ser possível armazenar todos os padrões de entrada na memória de um computador juntamente com os bits de classificação. O computador poderia então procurar pelo padrão desejado e responder com sua classificação.

O número de bits exigido para armazenar a mesma informação dos pesos perceptron pode ser substancialmente menor que o do método convencional da memória do computador, se a natureza dos padrões permitir uma representação compacta. Contudo, Minsky mostrou exemplos patológicos nos quais o número de bits exigidos para representar os pesos cresce exponencialmente com o tamanho do problema. Achar uma resposta para essa questão é uma área crítica das pesquisas de redes neurais.

2.2.4 - Aprendizado e Algoritmo de Treinamento

O aprendizado de perceptron é do tipo supervisionado. Uma rede perceptron é treinada pela apresentação de um conjunto de padrões para sua entrada, um de cada vez, e ajustando os pesos até que a saída desejada ocorra para cada uma das saídas.

O método de treinamento pode ser resumido como:

1- Aplicar um padrão na entrada e calcular a saída Y.

2- a. Se a saída esta correta, vá para o passo 1

b. Se a saída é incorreta, e é zero, adicione cada entrada ao seu peso correspondente; ou

c. Se a saída é incorreta, e é 1, subtraia cada entrada de seu peso correspondente

3- Vá para o passo 1

Uma questão que surge neste ponto, é como apresentar os padrões a rede. Devem ser aplicados seqüencialmente um após o outro ou selecionados aleatoriamente ? Existe pouca teoria para determinar qual opção seria melhor.

2.2.5 - A Regra Delta

Uma importante generalização do algoritmo de treinamento perceptron, chamada Regra Delta, estende esta técnica para entradas e saídas contínuas. Para ver como ela foi desenvolvida, note que o passo 2 do algoritmo de treinamento perceptron poderia ser generalizado pela introdução de um termo δ, o qual é a diferença entre a saída desejada (T) e a saída atual (A). Em símbolos:

$$\delta = (T - A) \tag{2.06}$$

O caso em que $\delta = 0$ corresponde ao passo 2a. O passo 2b corresponde a $\delta > 0$, enquanto o passo 2c corresponde a $\delta < 0$.

Em qualquer um desses casos, o algoritmo de treinamento perceptron é satisfeito se δ é multiplicado pelo valor de cada entrada (x_i) e este produto é adicionado ao correspondente peso. Para generalizar isto, um coeficiente "taxa de aprendizado" (***learning rate***) η multiplica o produto δx_i para permitir o controle do tamanho médio das mudanças dos pesos.

Simbolicamente:

$$\Delta_i = \eta \delta x_i \tag{2.07}$$

$$\omega_i(n+1) = \omega_i(n) + \Delta_i \tag{2.08}$$

onde:

Δi = a correção associada com a i-ésima entrada x_i

$\omega_i(n+1)$ = o valor do peso i depois do ajuste

$\omega_i(n)$ = o valor do peso i antes do ajuste

A Regra Delta modifica os pesos apropriadamente para as saídas alvo e atual de ambas as polaridades e para entradas e saídas contínuas ou binárias.

Não existe maneira de saber quantos passos serão necessários para que a rede aprenda. Isto está relacionado com a natureza do conjunto sendo aprendido.

2.3 - BACKPROPAGATION

2.3.1 - Introdução

A invenção do algoritmo backpropagation fez ressurgir grande parte do interesse em redes neurais artificiais. Backpropagation é um método sistemático para treinamento de redes neurais artificiais multicamadas. Apesar de suas limitações, backpropagation tem expandido dramaticamente a variação de problemas para os quais as redes neurais artificiais podem ser aplicadas.

2.3.2 - O Algoritmo de Treinamento Backpropagation

Configuração da rede

A figura 2.4 mostra o neurônio usado como bloco de construção fundamental para redes backpropagation. Um conjunto de entradas é aplicado do exterior ou de uma camada anterior. Cada uma delas é multiplicada por um peso, e os produtos são somados. Esta soma de produtos é chamada **NET** e deve ser calculada para cada neurônio na rede. Depois que **NET** é calculada, uma função de ativação **F** é aplicada para modifica-la, com isso produzindo o sinal **OUT**.

Existem muitas funções que podem ser usadas; o algoritmo backpropagation exige somente que a função seja diferenciável em qualquer ponto. A figura 2.5 mostra um exemplo destas funções.

A figura 2.6 mostra uma rede multicamadas adequada para treinamento com backpropagation. O primeiro conjunto de neurônios (conectados para as entradas) serve somente como pontos de distribuição (*fan-out*).

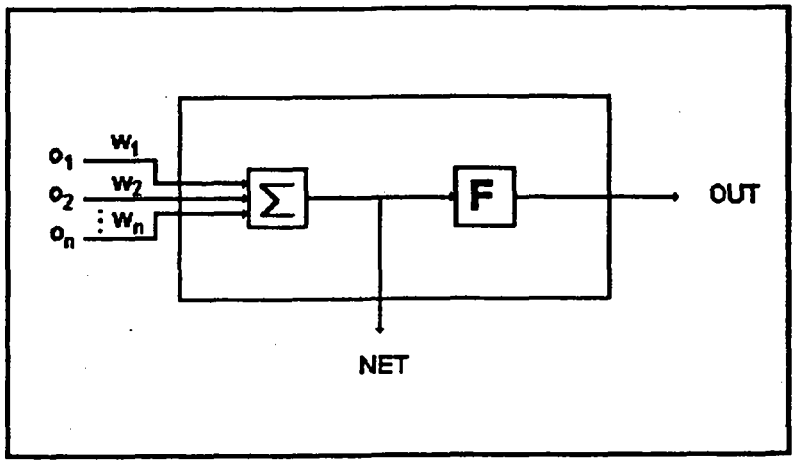

FIGURA 2.4 - Neurônio artificial com função de ativação

A figura 2.5 mostra a função de ativação usualmente usada para backpropagation.

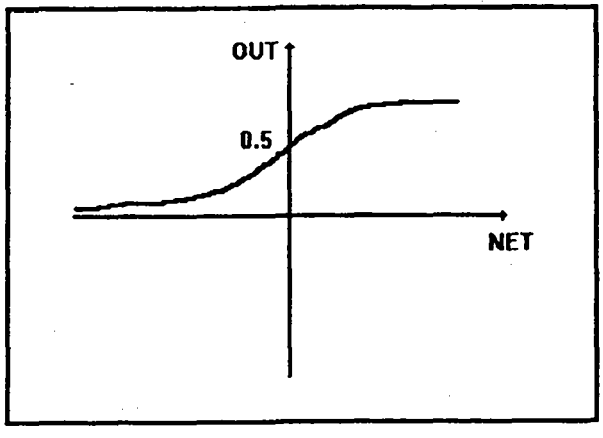

FIGURA 2.5 - Função de ativação sigmoidal

Backpropagation pode ser aplicado para redes com qualquer número de camadas.

Visão Geral do Treinamento

O objetivo do treinamento da rede é ajustar os pesos tal que a aplicação de um conjunto de entradas produz um conjunto desejável de saídas. Antes de iniciar o processo de

treinamento, todos os pesos devem ser inicializados para pequenos valores aleatórios. Isto assegura que a rede não estará saturada por grandes valores dos pesos e previne outras patologias do treinamento. Por exemplo, se todos os pesos iniciarem com valores iguais e o desempenho desejado exigir valores diferentes, a rede não irá aprender.

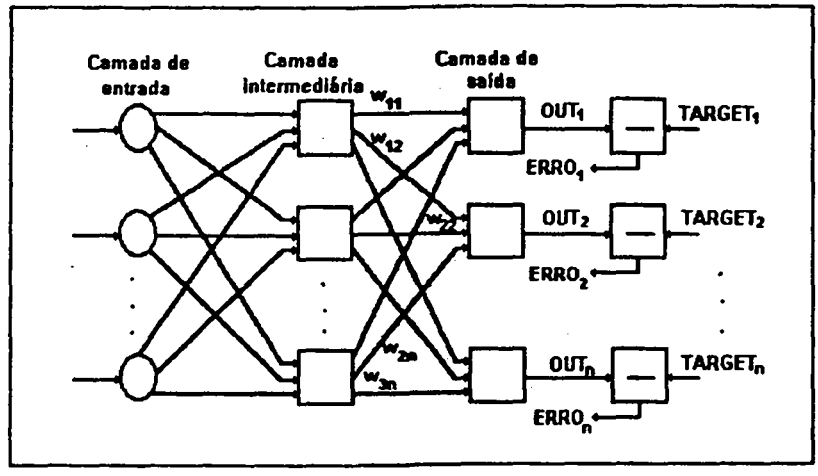

FIGURA 2.6 - Rede backpropagation de duas camadas

O treinamento da rede backpropagation exige os passos que seguem:

1 - Selecionar o próximo par de treinamento do conjunto de treinamento, aplicar o vetor de entrada para a entrada da rede.

2 - Calcular a saída da rede.

3 - Calcular o erro entre a saída da rede e a saída desejável (o vetor alvo do par) de treinamento.

4 - Ajustar os pesos da rede de uma maneira que minimize o erro.

5 - Repetir os passos de 1 a 4 para cada vetor no conjunto de treinamento até que o erro para todo conjunto seja aceitável.

Depois de um número de repetições suficiente dos 4 passos, o erro entre a saída atual e a saída alvo será reduzido para um valor aceitável, e a rede é dita ter sido treinada. Neste ponto, a rede é usada para reconhecimento e os pesos não são alterados.

Pode ser visto que os passos 1 e 2 constituem o ***forward pass*** no qual o sinal se propaga da entrada da rede para sua saída. Os passos 3 e 4 constituem o ***reverse pass***; daí, o sinal de erro calculado é propagado de volta até a rede onde ele é usado para ajustar os pesos.

- Forward Pass:

Produz a saída da rede, calculando as saídas intermediárias, camada por camada. Em notação vetorial, tem-se:

$$\mathbf{O} = \mathbf{F}(\mathbf{XW}) \tag{2.09}$$

onde:

O = vetor de saída

X = vetor de entrada

W = matriz de pesos

F = função de ativação

- Reverse Pass:

a) Ajuste dos pesos na camada de saída: como um valor alvo é disponível para cada neurônio na camada de saída, o ajuste dos pesos associados é facilmente realizado usando uma modificação da regra Delta, que é:

$$\delta = OUT(1-OUT)(target - OUT) \tag{2.10}$$

Então δ é multiplicado pelo OUT de um neurônio j, o neurônio fonte para o peso em questão. Este produto é novamente multiplicado por um coeficiente de taxa de treinamento η e o resultado é adicionado ao peso anterior para obter o novo valor. Um processo idêntico é efetuado para cada peso procedendo de um neurônio na camada interna para um na camada de saída.

As equações seguintes ilustram este cálculo:

$$\Delta\omega_{pqk} = \eta\delta_{qk}OUT_{pj} \tag{2.11}$$

$$\omega_{pqk}(n+1) = \omega_{pqk}(n) + \Delta\omega_{pqk} \tag{2.12}$$

onde,

$\omega_{pqk}(n)$ = o valor de um peso do neurônio **p** na camada intermediária para um neurônio **q** na camada de saída no passo **n** (antes do ajuste); note que o índice **k** indica que o peso é associado com sua camada de destinação.

$\omega_{pqk}(n+1)$ = valor do peso no passo **n+1** (depois do ajuste).

δ_{qp} = o valor de δ para o neurônio **q** na camada de saída **k**

OUT_{pj} = o valor de OUT para o neurônio **p** na camada intermediária **j**.

Note que os índices **p** e **q** referem-se à um neurônio específico, enquanto os índices **j** e **k** referem-se a uma camada.

b) Ajuste dos pesos de camadas intermediárias (hidden layer): as camadas intermediárias não têm vetor alvo, então o processo de treinamento descrito acima não pode ser usado. Backpropagation treina as camadas intermediárias pela propagação do erro da saída de volta através da rede, camada por camada, ajustando os pesos em cada camada.

As equações (2.11) e (2.12) são usadas para todas as camadas, tanto de saída quanto as intermediárias; contudo, para as camadas intermediárias, δ deve ser gerado sem o uso de um vetor alvo. Primeiro, δ é calculado para cada neurônio na camada de saída, como anteriormente. Ele é usado para ajustar os pesos alimentando a camada de saída, então ele é propagado de volta através dos mesmo pesos gerando um valor para δ para cada neurônio na primeira camada intermediária. Esses valores de δ são usados para ajustar os pesos desta camada intermediária e de uma maneira similar, são propagados de volta para todas as camadas precedentes.

2.4 - COUNTERPROPAGATION

2.4.1 - Introdução

A rede counterpropagation desenvolvida por Robert Hecht-Nielsen (apud [1]) vai além dos limites de redes de uma camada. Quando comparada com backpropagation, ela pode

reduzir o tempo de treinamento em uma centena de vezes. Counterpropagation não é tão geral quanto backpropagation, mas providencia uma solução para aquelas aplicações que não podem tolerar longas sessões de treinamento.

Counterpropagation é a combinação de dois algoritmos bem conhecidos: o ***self-organizing map*** de Kohonen e o ***outstar*** de Grossberg [2].

Métodos tais como counterpropagation que combinam paradigmas de redes numa visão de construção em blocos podem produzir redes mais próximas da arquitetura do cérebro humano que qualquer estrutura homogênea.

O processo de treinamento associa vetores de entradas com correspondentes vetores de saída. Esses vetores podem ser binários, consistindo de zeros e uns, ou contínuos. Uma vez que a rede está treinada, a aplicação de um vetor de entrada produz o vetor de saída desejado. A capacidade de generalização da rede permite que ela produza uma saída correta, mesmo quando é dado um vetor de entrada parcialmente incompleto ou parcialmente incorreto. Isto torna a rede útil para reconhecimento de padrões, reconstituição de padrões e aplicações de melhoramentos de sinais.

2.4.2 - Estrutura da Rede

A figura 2.7 mostra uma versão simplificada da rede counterpropagation; ela ilustra as características funcionais deste paradigma.

Os neurônios na camada 0 (mostrados como círculos) servem somente como pontos de distribuição (***fan-out***) e não efetuam nenhuma computação. Cada neurônio na camada 0 conecta-se com todos os neurônios na camada 1 (chamada camada Kohonen) através de um peso w_{mn}; eles serão coletivamente referidos como a matriz de pesos **W**. Similarmente, cada neurônio na camada Kohonen (camada 1) conecta-se com todos os neurônios na camada Grossberg (camada 2) por um peso v_{np}; eles compreendem a matriz de pesos **V**. Esta estrutura se assemelha a outras redes vistas anteriormente; contudo, a diferença está no processamento feito nos neurônios Kohonen e Grossberg.

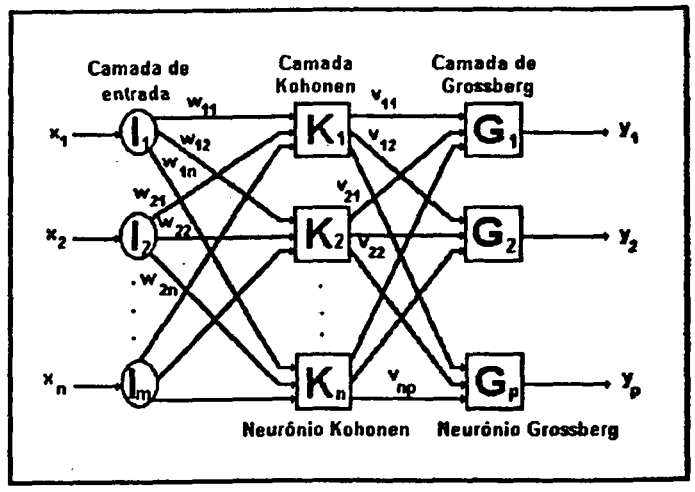

FIGURA 2.7 - Rede counterpropagation

2.4.3 - Modos de Operação e Treinamento

Como muitas outras redes, counterpropagation funciona em dois modos: o modo normal, no qual ela aceita um vetor de entrada **X** e produz um vetor de saída **Y**, e o modo de treinamento, no qual um vetor de entrada é aplicado e os pesos são ajustados de modo a obter-se o vetor de saída desejado.

a) Operação Normal - Camada Kohonen:

Na sua forma mais simples a camada de Kohonen funciona numa visão "ganhador leva tudo" (*"winner-takes-all"*); isto é, para um dado vetor de entrada, uma e somente uma saída dos neurônios Kohonen é logicamente 1; todas as outras são zero.

Associado com cada neurônio Kohonen está um conjunto de pesos conectando-o a cada vetor de entrada. Por exemplo, na figura 2.7, o neurônio Kohonen K_1 tem pesos $w_{11}, w_{21}, \ldots\ldots w_{m1}$, compreendendo um vetor de pesos $\mathbf{W}_1$. Como com neurônios em muitas redes, a saída NET de cada neurônio Kohonen é simplesmente a somatória das entradas ponderadas. Isto pode ser expressado como segue:

$$NET_j = w_{1j}x_1 + w_{2j}x_2 + \ldots + w_{mj}x_m \qquad (2.13)$$

onde NET_j é saída NET do neurônio Kohonen j

$$NET_j = \sum_i x_i w_{ij} \tag{2.14}$$

ou em notação vetorial:

$$\mathbf{N} = \mathbf{XW} \tag{2.15}$$

onde **N** é o vetor de saída NET da camada Kohonen.

O neurônio Kohonen com maior valor NET é o "ganhador". Sua saída é inicializada para 1; todas as outras são inicializadas para zero.

b) Operação Normal - Camada Grossberg:

A camada Grossberg funciona de maneira similar. Sua saída NET é a somatória ponderada das saídas da camada Kohonen K_1, K_2, ...,K_n, formando o vetor **K**. A saída NET de cada neurônio Grossberg é então

$$NET_j = \sum_i k_i v_{ij} \tag{2.16}$$

onde NET_j é a saída do neurônio Grossberg j, ou em forma vetorial

$$\mathbf{Y} = \mathbf{KV} \tag{2.17}$$

onde:

Y = vetor de saída da camada Grossberg

K = vetor de saída da camada Kohonen

V = matriz de pesos da camada Grossberg

Se a camada Kohonen é operada tal que somente uma saída NET é 1 e todas as outras são zero, somente um elemento do vetor **K** não é zero, e o cálculo é simples. De fato, a única ação de cada neurônio na camada Grossberg é colocar na saída o valor do peso que conecta-o com o único neurônio Kohonen cuja saída não é zero.

c) Modo de Treinamento - Treinamento da Camada Kohonen:

A camada Kohonen classifica os vetores de entrada dentro de grupos que são similares. Isto é conseguido com a ajuste dos pesos da camada Kohonen tal que vetores de entrada similares ativam o mesmo neurônio Kohonen. Daí em diante é responsabilidade da camada Grossberg produzir a saída desejada.

O treinamento Kohonen é um algoritmo auto-organizador (***self-organizing***) que opera em modo não supervisionado. Por esta razão, é difícil (e desnecessário) predizer qual neurônio Kohonen específico será ativado por um dado vetor de entrada. Só é necessário assegurar que o treinamento separe os vetores de entrada que não sejam similares.

Para o treinamento da camada Kohonen, um vetor de entrada é aplicado e seu produto com o vetor de pesos associado é calculado para cada neurônio Kohonen. O neurônio com maior produto é declarado o "ganhador" e seus pesos são ajustados. Como a operação produto usada para calcular os valores de NET é uma medida de similaridade entre os vetores de entrada e peso, o processo de treinamento consiste na seleção do neurônio Kohonen cujo vetor peso é mais similar ao vetor de entrada, e torná-lo ainda mais similar. Note novamente que este treinamento é não supervisionado. A rede se auto-organiza tal que um dado neurônio Kohonen tenha saída máxima para um dado vetor de entrada. A equação de treinamento que é usada é a seguinte:

$$w_{novo} = w_{velho} + \alpha(x - w_{velho}) \tag{2.18}$$

onde:

w_{novo} = o novo valor de um peso conectando um componente de entrada x para o neurônio ganhador.

w_{velho} = o valor anterior deste peso.

α = um coeficiente de taxa de treinamento que pode variar durante o processo de treinamento.

Cada peso associado com o neurônio Kohonen vencedor é mudado por uma quantidade proporcional à diferença entre seu valor e o valor da entrada com a qual ele é conectado. A direção da mudança minimiza a diferença entre o peso e sua entrada.

A figura 2.8 mostra este processo geometricamente em duas dimensões. Primeiro, o vetor $\mathbf{X} - \mathbf{W}_{velho}$ é achado pela construção de um vetor que liga o final de **W** ao final de **X**. A seguir, este vetor é multiplicado pelo escalar α, um número menor que um, com isso produzindo o vetor de mudanças δ. Finalmente, o novo vetor de pesos $\mathbf{W}_{novo}$ é uma linha da origem para o final de δ. Isto mostra que o efeito do treinamento é rotacionar o vetor peso em direção ao vetor de entrada sem materialmente mudar seu comprimento.

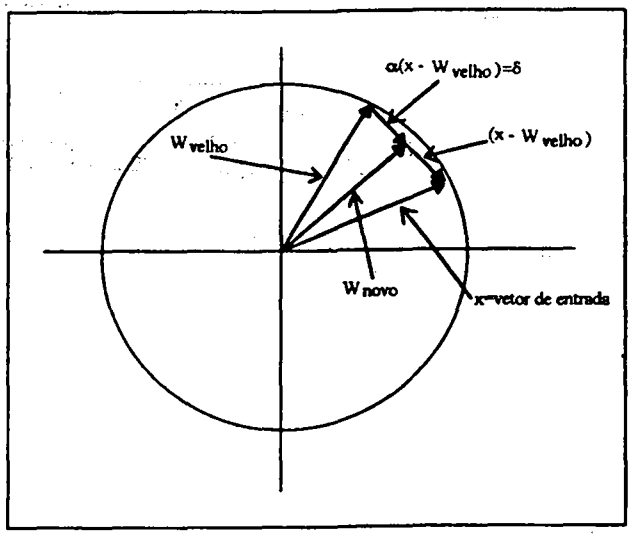

FIGURA 2.8 - Cálculo do vetor peso geometricamente

A variável α é um coeficiente de taxa de treinamento que usualmente inicia próximo a 0.7 [1] e pode ser reduzido gradualmente durante o treinamento. Isto permite grandes passos iniciais e pequenos passos quando o treinamento esta no final.

Se somente um vetor de entrada fosse associado com cada neurônio Kohonen, a camada Kohonen poderia ser treinada com um cálculo por peso. Os pesos de um neurônio ganhador poderiam ser feitos igual aos componentes do vetor de treinamento ($\alpha = 1$). Geralmente o conjunto de treinamento inclui muitos vetores de entrada que são similares e a rede deve ser treinada para ativar o mesmo neurônio Kohonen para cada um deles. Neste caso, os pesos deste neurônio devem ser a média dos vetores de entrada que irão ativá-los. Inicializando α para valores

baixos irá reduzir o efeito de cada passo do treinamento, fazendo o valor final uma média dos vetores de entrada para o qual ele foi treinado. Desta maneira, os pesos associados com um neurônio irão assumir um valor próximo ao "centro" dos vetores de entrada para o qual aquele neurônio é o "ganhador ".

- Inicializando os Vetores Peso

Os pesos devem ser ajustados para valores iniciais antes do início do treinamento. É prática comum em redes neurais randomizar os pesos para pequenos valores. Para o treinamento Kohonen, os vetores pesos devem ser normalizados.

Randomizar os pesos da camada Kohonen pode causar sérios problemas no treinamento, porque os vetores pesos serão uniformemente distribuídos ao redor da hiperesfera. Como os vetores de entrada não são geralmente uniformemente distribuídos e tendem a ficar agrupados numa porção relativamente pequena da superficie da hiperesfera, muitos dos vetores peso ficarão muito longe de qualquer vetor de entrada, de forma que eles nunca serão de fato utilizados. Esses neurônios Kohonen terão sempre a saída igual a zero e serão desperdiçados. Além disso, os pesos restantes podem ser poucos para permitir separar os vetores de entrada em categorias.

Suponha que existem vários conjuntos de vetores de entrada, todos os quais são similares, mas ainda devem ser separados dentro de diferentes categorias. A rede deve ser treinada para ativar um neurônio Kohonen diferente para cada categoria. Se a densidade inicial dos vetores pesos é muito baixa nas vizinhanças dos vetores de treinamento, pode ser impossível separar categorias similares, por não existirem vetores pesos suficientes na vizinhança para designar um para cada categoria de vetor de entrada.

Inversamente, se vários vetores de entrada são pequenas variações do mesmo padrão e devem ser agrupados juntos, eles devem disparar um único neurônio Kohonen. Se a densidade dos vetores pesos é muito alta próximo a um grupo de vetores de entrada ligeiramente diferentes, cada entrada pode ativar um neurônio Kohonen diferente. Isto não é catastrófico, porque a

camada Grossberg pode mapear diferentes neurônios Kohonen para a mesma saída, mas é um desperdício de neurônios Kohonen.

A solução mais desejável é distribuir os vetores pesos de acordo com a densidade de vetores de entrada que devem ser separados, colocando mais vetores pesos na vizinhança de um grande número de vetores de entrada. É impraticável implementar isto diretamente, mais várias técnicas aproximam este efeito.

Uma solução, chamada o "Método da Combinação Convexa", inicializa todos os pesos com o mesmo valor $1/\sqrt{n}$, onde **n** é o número de entradas, que corresponde ao número de componentes em cada vetor peso. Isto faz com que os vetores peso tenham comprimento unitário e sejam coincidentes. Além disso, a cada componente x_i é dado um valor $\alpha x_i + \{[1/\sqrt{n}](1-\alpha)\}$, onde **n** é o número de entradas. Inicialmente é dado a α um valor pequeno, fazendo com que todos os vetores de entrada tenham um comprimento próximo a $1/\sqrt{n}$ e sejam coincidentes com os vetores pesos. Durante o treinamento, α é gradualmente incrementado até 1. Isto permite os vetores de entrada serem separados e eventualmente assumirem os valores originais. Os vetores pesos seguem um ou um pequeno grupo de vetores de entrada e no final do processo de treinamento é produzido o padrão de saída desejado. O método da combinação convexa opera bem, mas torna lento o processo de treinamento.

Outro método adiciona ruído aos vetores de entrada. Isto faz com eles movam-se aleatoriamente, eventualmente capturando um vetor peso. Este método também trabalha bem, mas é ainda mais lento que a combinação convexa.

Um terceiro método inicia randomizando os pesos, num estágio inicial do processo de treinamento, ajusta todos os pesos, inclusive aqueles que não estão relacionados com o neurônio Kohonen vencedor. Isto move os vetores peso ao redor da região dos vetores de entrada. Com o progresso do processo de treinamento, o ajuste dos pesos é restringido para aqueles neurônios Kohonen que estão mais próximos do vencedor. Este raio de ajuste é gradualmente diminuído até que os únicos pesos a serem ajustados sejam aqueles associados com o neurônio Kohonen vencedor.

Existe ainda outro método que dá a cada neurônio Kohonen uma "consciência". Se ele tem vencido mais vezes que o necessário (grosseiramente 1/k, onde **k** é o número de neurônios Kohonen), ele é temporariamente suspenso a um valor que reduz a sua chance de vencer, desse modo dando aos outros neurônios uma oportunidade de serem treinados [1].

Em muitas aplicações, o problema da distribuição dos pesos pode afetar seriamente a precisão dos resultados. Infelizmente, os efeitos das várias soluções ainda não foram totalmente avaliados.

- Modo Interpolativo

Até aqui tem-se discutido um algoritmo de treinamento no qual somente o neurônio Kohonen é ativado por cada vetor de entrada; isto é chamado *modo acrescentativo* (***accretive mode***). A precisão deste método está limitada para aquela saída que é totalmente uma função de um único neurônio Kohonen.

No *modo interpolativo*, um grupo de neurônios Kohonen tendo as maiores saídas apresentam suas saídas para a camada Grossberg. O número de neurônios neste grupo deve ser escolhido para a aplicação, e não existem evidências conclusivas de qual tamanho seria ótimo. Uma vez que o grupo é determinado seu conjunto de saídas NET como um vetor é normalizado para um comprimento unitário pela divisão de cada valor NET pela raiz quadrada do somatório dos quadrados dos valores NET no grupo. Todos os neurônios fora do grupo têm sua saída zeradas.

O modo interpolativo é capaz de representar mapeamentos mais complexos e pode produzir resultados mais precisos. Novamente, nenhuma evidência conclusiva é disponível para se avaliar os modos interpolativo e acrescentativo.

- Propriedades Estatísticas do Treinamento da Rede

O treinamento Kohonen tem uma útil e interessante habilidade para extrair as propriedades estatísticas do conjunto de dados de entrada. Kohonen (apud [1]) mostrou que, em uma rede totalmente treinada, a probabilidade de um vetor de entrada selecionado

randomicamente (selecionado de acordo com a função densidade de probabilidade do conjunto de entrada), melhor aproxima-se a qualquer vetor peso é 1/k, onde k é o número de neurônios Kohonen. Esta é a distribuição ótima dos pesos na hiperesfera. Isto assume que todos os vetores pesos estão em uso, uma situação que se realizará somente se um dos métodos discutidos é usado para distribuir os vetores de peso.

d) Modo de Treinamento - Treinamento da Camada Grossberg:

A camada Grossberg é relativamente simples de treinar. Um vetor de entrada é aplicado, as saídas Kohonen são estabelecidas, e a saída Grossberg é calculada como na operação normal. A seguir, cada peso é ajustado somente se ele esta conectado a um neurônio Kohonen que tem uma saída diferente de zero. A quantidade de ajuste no peso é proporcional a diferença entre o peso e a saída desejada do neurônio Grossberg com o qual ele esta conectado. Em símbolos

$$v_{ijnovo} = v_{ijvelho} + \beta(y_j - v_{ijvelho})k_i \qquad (2.19)$$

onde

k_i = a saída do neurônio Kohonen i (somente os neurônios diferentes de zero).

y_i = componente j do vetor de saídas desejadas.

Inicialmente β é inicializado em aproximadamente 0.1 [1] e é gradualmente reduzido com o progresso do treinamento.

Com isso pode-se ver que os pesos da camada Grossberg irão convergir para valores médios das saídas desejadas, considerando que os pesos da camada Kohonen são treinados para valores médios das entradas. O treinamento Grossberg é supervisionado; o algoritmo tem a saída desejada para o qual ele é treinado. A não supervisionada operação de auto-organização da camada Kohonen produz saídas em posições indeterminadas; estas são mapeadas para as saídas desejadas pela camada Grossberg.

2.4.4 - A Rede Totalmente Counterpropagation

A figura 2.9 mostra a rede totalmente counterpropagation. Em operação normal, os vetores de entrada **X** e **Y** são aplicados e a rede treinada produz os vetores de saídas **X'** e **Y'**, os quais são aproximações de **X** e **Y**, respectivamente. Neste caso assume-se que **X** e **Y** são vetores unitários normalizados; daí, eles tenderão a produzir vetores normalizados na saída.

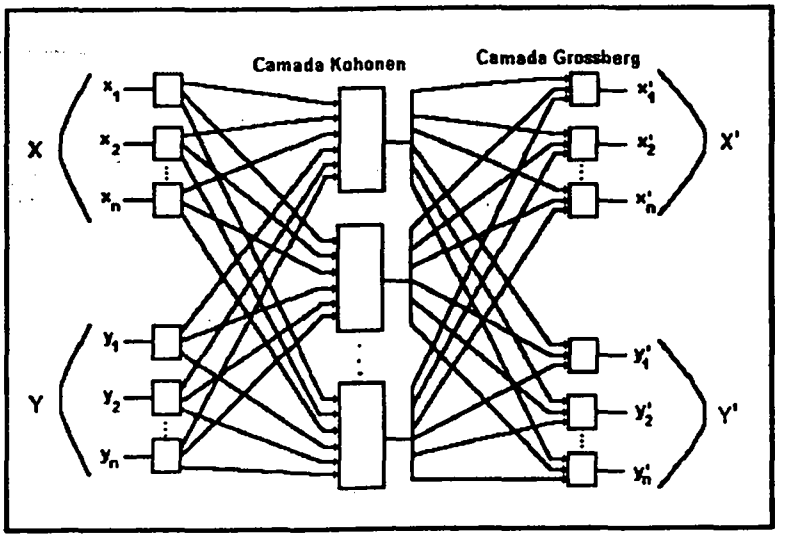

FIGURA 2.9 - Rede totalmente counterpropagation

Durante o treinamento, os vetores **X** e **Y** são aplicados respectivamente como entradas da rede e saídas desejadas. **X** é usado para treinar as saídas **X'**, enquanto **Y** é usado para treinar as saídas **Y'** da camada Grossberg. A rede totalmente counterpropagation é treinada usando o mesmo método descrito para a rede anterior. Os neurônios Kohonen recebem entradas dos vetores **X** e **Y**.

O resultado é um mapeamento idêntico no qual a aplicação de um par de vetores de entrada produz suas replicas na saída. Isto não parece muito interessante até que se imagina que aplicando somente o vetor **X** (com o vetor **Y** inicializado em zero) produz-se os vetores de saída **X'** e **Y'**. Se F é uma função de mapeamento de **X** para **Y'**, então a rede aproxima-se dela. Também, se o inverso de F existe, aplicando somente o vetor **Y** (atribuindo **X** para 0) produz-se

X'. Esta habilidade única para gerar uma função e sua inversa torna a rede counterpropagation útil em numerosas aplicações.

2.4.5 - Conclusão

Robert Hecht-Nielsen, o inventor da rede counterpropagation (CPN), cita suas limitações:" CPN é obviamente inferior a Backpropagation para muitas aplicações de mapeamento de rede. Suas vantagens são que ela é simples e proporciona um bom modelo estatístico de seu ambiente de vetor de entrada" [1].

Deve-se também observar a rapidez do treinamento; apropriadamente aplicado ele pode melhorar em muito o tempo de computação. Ela também é útil para sistemas de rápida prototipagem, onde a grande precisão de Backpropagation faz dela o método escolhido na versão final, mas onde uma rápida aproximação é importante. Além disso, a capacidade de generalizar uma função e sua inversa é de grande utilidade em inúmeros sistemas.

2.5 - MÉTODOS ESTATÍSTICOS DE TREINAMENTO

2.5.1 - Introdução

Métodos estatísticos são úteis tanto para o treinamento de uma rede neural artificial quanto para produzir uma saída de uma rede previamente treinada. Métodos estatísticos de treinamento oferecem importantes vantagens, como por exemplo, evitar mínimos locais no treinamento, mas eles também tem suas limitações.

2.5.2 - Aplicações de Treinamento

Uma rede neural artificial é treinada por meio de algum processo que modifica seus pesos. Se o treinamento tem sucesso, a aplicação de um conjuntos de entradas produz o desejado conjunto de saídas. Existem duas categorias de métodos de treinamento: determinístico e estatístico. Um ***método de treinamento determinístico*** segue um procedimento passo a passo para ajustar os pesos da rede baseados no seus valores correntes e nos valores das entradas,

saídas atuais e saídas desejadas. O treinamento perceptron é um exemplo do treinamento determinístico.

Métodos estatísticos de treinamento fazem mudanças pseudo aleatórias nos valores dos pesos, retendo aquelas mudanças que resultam em melhoramentos.

O objetivo do treinamento é minimizar a diferença entre a saída da rede e a saída desejada, freqüentemente chamada de *função objetivo*.

É feita uma seleção aleatória de um peso, que deve ser ajustado por um pequeno valor aleatório. Se o ajuste reduz a função objetivo, o novo peso é mantido; senão, o peso é reposto em seu valor anterior.

Este processo tende a minimizar a função objetivo, mas pode levar a uma solução pobre. A figura 2.10 mostra como isso pode ocorrer num sistema de um único peso. Assuma que o peso é ajustado inicialmente no ponto A. Se os passos aleatórios dos pesos são pequenos, todas as derivações irão incrementar a função objetivo e serão rejeitadas. Se os passos aleatórios dos pesos são grandes, ambos os pontos A e B serão visitados freqüentemente; mas as mudanças serão tão drásticas que nunca se encontrara o mínimo desejado. Uma estratégia útil para evitar esses problemas é iniciar com grandes passos e gradualmente reduzir o tamanho médio dos passos. Isto permite que a rede encontre um mínimo local, enquanto assegura uma eventual estabilização da rede.

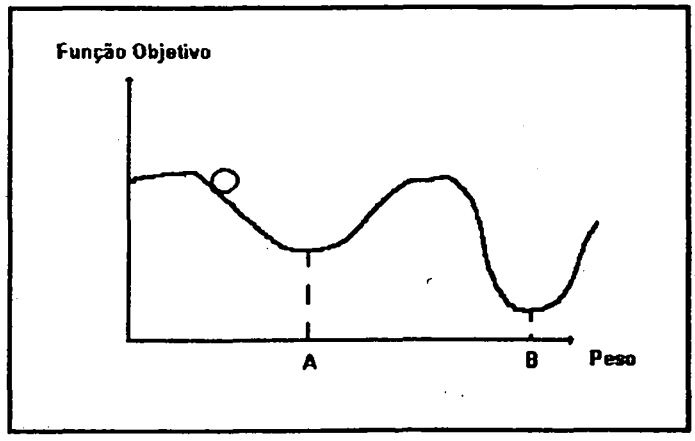

FIGURA 2.10 - Problemas de mínimo local

Este procedimento tem uma forte semelhança com a destempera de metais; daí , o termo "*simulated annealing*" ser freqüentemente usado para descrevê-lo. No processo de destempera, a distribuição dos estados de energia é determinada pelo relacionamento que segue:

$$P(e)\alpha \exp(-e/kT) \tag{2.20}$$

onde:

P(e) = probabilidade que o sistema esteja em um estados com energia **e**

k = constante de Boltzmann

T = temperatura (graus Kelvin)

2.5.3 - Treinamento Boltzmann

A aplicação deste método estatístico para treinamento de redes neurais é como segue:

1 - Define-se uma variável T que representa uma temperatura inicial com T no maior valor

2 - Aplica-se um conjunto de entradas para a rede, e calcula-se a saída e a função objetivo

3 - Faz-se uma mudança aleatória no peso e calcula-se novamente a saída da rede e a mudança na função objetivo devido a mudança no peso.

4 - Se a função objetivo é reduzida (melhorada), retenha a mudança no peso.

Se a mudança no peso resulta em um incremento na função objetivo, calcule a probabilidade de aceitar essa mudança da distribuição de Boltzmann como segue:

$$P(c) = \exp(-c/kT) \tag{2.21}$$

onde:

P(c) = a probabilidade de uma mudança de **c** na função objetivo

k = uma constante análoga a constante de Boltzmann que deve ser escolhida para o problema

T = temperatura artificial

Selecione um número aleatório **r** de uma distribuição uniforme entre zero e um. Se P(c) é maior que **r**, retenha a mudança; senão retorne o valor do peso para o valor anterior.

2.5.4 - Treinamento Cauchy

Szu e Hartley (apud [1]) desenvolveram um método para treinamento rápido desses sistemas. Este método substitui a distribuição Boltzmann pela distribuição Cauchy quando calcula o tamanho do passo. Com essa simples substituição, a taxa de redução da temperatura máxima é feita inversamente linear, preferível que inversamente logarítmica, como no algoritmo de treinamento Boltzmann. Isto reduz drasticamente o tempo de treinamento. Esta relação pode ser expressa como segue:

$$T(t) = T_0/(1+t) \tag{2.22}$$

onde:

T(t) = temperatura artificial como uma função do tempo

T_O = temperatura artificial inicial

t = tempo artificial

A distribuição Cauchy é:

$$P(x) = T(t)/[T(t)^2 + x^2] \tag{2.23}$$

onde P(x) é a probabilidade de um passo de tamanho x.

P(x) pode ser integrado por métodos usuais. Daí obtêm-se **x** pela expressão:

$$x_i = \rho\{T(t)\mathrm{Tan}[P(x)]\} \tag{2.24}$$

onde:

ρ = coeficiente taxa de treinamento

x_i = a mudança do peso

2.5.5 - Método do Calor Específico Artificial

A despeito do melhoramento do método Cauchy, os tempos de treinamento ainda permanecem longos. Uma técnica usada em termodinâmica pode ser usada para acelerar este processo. Este método envolve o ajuste da taxa de redução de temperatura de acordo com um "calor específico" artificial calculado durante o processo de treinamento.

2.5.6 - Backpropagation e Treinamento Cauchy

Backpropagation tem a vantagem de uma busca direta: isto é, os pesos são sempre ajustados na direção que minimiza a função erro. Enquanto o tempo de treinamento é freqüentemente grande, ele é consideravelmente mais rápido que o método de busca aleatória da máquina Cauchy, o qual acha um mínimo global, mas pode tomar muitos passos incorretos e grandes tempos de treinamento.

Combinando as duas técnicas produziu-se resultados encorajadores. Fazendo o ajuste dos pesos ser igual à soma do calculado pelo algoritmo backpropagation e o passo aleatório exigido pelo algoritmo Cauchy, produz-se um sistema que converge e acha o mínimo global mais rapidamente que um sistema treinado por só um dos métodos. Uma heurística simples é usada para evitar que o aprendizado da rede paralise [1], um problema que pode ocorrer em ambos os método de treinamento.

2.5.7 - Problemas com Backpropagation

Apesar da potencialidade de backpropagation, alguns problemas podem surgir em suas aplicações, contudo alguns foram reduzidos com o uso de novos algoritmos.

- Convergência

Rumelhart, Hinton e Williams (apud[1]) apresentaram uma prova da convergência em termos de equações diferenciais parciais, que só é válida se os pesos da rede são ajustados com passos infinitesimais. Como isso implica em um tempo de convergência infinito, esta prova não diz nada a respeito de tempos de treinamento em aplicações práticas. De fato, não existe prova

que backpropagation irá mesmo convergir com um passo de tamanho finito. Observações empíricas mostram que a rede usualmente treina, mas que a duração do processo não é previsível e longa.

- Mínimo Local

Backpropagation usa gradiente descendente para ajustar os pesos da rede, seguindo a inclinação da superfície de erro para um mínimo. Isto trabalha bem com superfícies de erro convexas, as quais têm um único mínimo, mas geralmente conduz para uma solução que não é ótima com superfícies altamente convoluídas e não convexas encontradas em problemas práticos. Em alguns casos, um mínimo local é aceitável, em outros não.

Mesmo depois que a rede foi treinada, não existe maneira de saber se ela encontrou um mínimo global. Se a solução não é satisfatória, a solução é reinicializar os pesos com novos valores aleatórios e retreinar a rede, sem garantias de que será encontrada uma solução aceitável, mesmo se for encontrado um mínimo global.

- Paralisação

Sob certas circunstâncias, uma rede pode atingir um estado no qual as modificações nos pesos conduzem a uma paralisação virtual. Esta "paralisação da rede" é um sério problema: uma vez neste estado, o tempo de treinamento da rede pode ser bastante aumentado

Paralisação ocorre quando grande percentagem dos neurônios alcançam valores tão grandes que produzem valores altos para NET. Isto faz OUT aproximar-se de seu limite. Neste ponto a derivada da função sigmóide aproxima-se de zero. Como já foi visto, o algoritmo backpropagation calcula a magnitude das mudanças nos pesos usando esta derivada como um fator na expressão. Para os neurônios afetados, a derivada próximo a zero faz as mudanças nos pesos aproximar-se de zero; se isto se expandir para quase toda a rede, o treinamento ficará tão lento que irá parecer ter parado.

Não existe teoria para predizer se uma rede irá sofrer paralisação durante o treinamento, ou não. Experimentalmente, pequenos tamanhos de passos têm causado a

paralisação ser menos freqüente, mas um passo que é pequeno para um problema pode ser excessivo para outro. O custo da paralisação pode ser alto. Simulações podem consumir muitas horas de tempo de computação, somente para terminar numa falha de treinamento (paralisação).

2.5.8 - Problemas com o Algoritmo de Treinamento Cauchy

Apesar do melhoramento na taxa de treinamento providenciado pela máquina Cauchy quando comparado com a máquina de Boltzmann, o tempo de convergência pode ser ainda 100 vezes aquele do algoritmo Backpropagation. Também, a paralisação na rede é muito severa usando o algoritmo de treinamento Cauchy, especialmente em uma rede com não-linearidade, tais como uma função logística. A variação infinita da distribuição Cauchy permite mudanças nos pesos de magnitude não limitada. Além disso, grandes mudanças nos pesos algumas vezes serão aceitáveis embora elas sejam desvantajosas, pois freqüentemente resultam em severa saturação dos neurônios na rede, com risco de paralisação.

2.6 - HOPFIELD

2.6.1 - Introdução

As redes apresentadas nos capítulos anteriores são não recorrentes, isto é, não existe realimentação da saída das redes para as suas entradas. A falta de realimentação assegura que as redes são incondicionalmente estáveis. Elas não podem entrar em um modo no qual a rede vai interminavelmente de um estado a outro, sem produzir uma saída útil. Esta característica desejável vem com um preço: redes não recorrentes têm um repertório de comportamentos limitado comparado com uma rede recorrente equivalente.

Ainda não é possível predizer quais redes são estáveis e quais aquelas que mudam continuamente. Além disso, o problema parece tão difícil que alguns pesquisadores foram pessimistas a respeito da possibilidade de se achar uma solução. Felizmente, um teorema que define um subconjunto das redes recorrentes cuja saída eventualmente alcança um estado estável foi demonstrado por Cohen e Grossberg (apud [1]). Este fato abriu as portas para muitas

pesquisas, e hoje muitos cientistas estão explorando o complicado comportamento e capacidades dessas redes.

John Hopfield teve uma importante contribuição tanto para a teoria quanto para aplicação das redes recorrentes. Como resultado, algumas configurações ficaram conhecidas como redes Hopfield.

2.6.2 - Configuração de Redes Recorrentes

A figura 2.11 mostra uma rede recorrente consistindo de duas camadas. O formato é um pouco diferente daquele encontrado no trabalho de Hopfield, contudo, ele é funcionalmente equivalente as redes apresentadas nos itens anteriores.

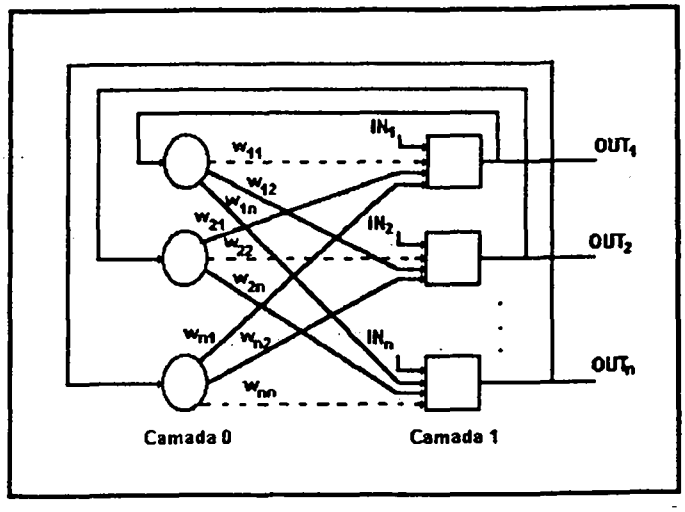

FIGURA 2.11 - Rede recorrente de duas camadas

2.6.3 - Sistemas Binários

Nos primeiros trabalhos de Hopfield (apud [1]), a função de ativação F era um limite. A saída de tais neurônios é 1 se a somatória das saídas ponderadas dos outros neurônios é maior que um limite T_j; de outro modo é 0. Isto é calculado como segue:

$$NET_j = \sum_{i=j} w_{ij} OUT_i + IN_j \tag{2.25}$$

$$OUT_j = 1 \text{ se } NET_j > T_j \tag{2.26}$$

$$OUT_j = 0 \text{ se } NET_j < T_j \tag{2.27}$$

OUT_j fica inalterado se $NET_j = T_j$

O *estado de uma rede* é simplesmente o conjunto dos valores correntes dos sinais OUT de todos os neurônios. Na rede Hopfield original, o estado de cada neurônio muda em intervalos aleatórios discretos no tempo; num trabalho posterior, os estados podiam mudar simultaneamente. Como a saída de um neurônio binário pode ser somente um ou zero (níveis intermediários não são permitidos), o estado corrente da rede forma um número binário, cada bit representa um sinal OUT de um neurônio.

2.6.4 - Estabilidade

Como em outras redes, os pesos entre as camadas da rede podem ser considerados na forma de matriz **W**. Cohen e Grossberg (apud [1]) mostraram que redes recorrentes são estáveis se a matriz é simétrica com zeros na diagonal principal; isto é, se $w_{ij} = w_{ji}$ para i diferente de j, e $w_{ii} = 0$ para todo i.

A estabilidade dessas redes pode ser provada através de técnicas matemáticas. Suponha uma função que decrementa cada vez que a rede muda de estado. Eventualmente esta função deve alcançar um mínimo e parar, com isso assegurando que a rede é estável. A função que segue é chamada de função Liapunov e trabalha exatamente da maneira das redes recorrentes apresentadas acima.

$$E = (-1/2)\sum_i \sum_j w_{ij} OUT_i OUT_j - \sum_j I_j OUT_j + \sum_j T_j OUT_j \qquad (2.28)$$

onde:

E = energia da rede artificial

w_{ij} = peso da saída do neurônio i para a entrada do neurônio j.

OUT_j = saída do neurônio j

I_j = entrada externa do neurônio j

T_j = limite do neurônio j

A mudança na energia E, devido a mudança no estado do neurônio j, é

$$\delta E = -\left[\sum_{i \neq j}\left(w_{ij}OUT_i\right) + I_j - T_j\right]\delta OUT_j \tag{2.29}$$

$$\delta E = -\left[NET_j - T_j\right]\delta OUT_j \tag{2.30}$$

onde δOUT_j é a mudança na saída do neurônio j.

Suponha que o valor NET do neurônio é maior que o limite. Isto faz com que o termo entre colchetes seja positivo, e da equação 2.25, a saída do neurônio j deve mudar na direção positiva (ou permanecer constante). Isto significa que δ OUT_j pode ser somente positivo ou zero, e δ E deve ser negativo; daí, a energia da rede deve ou diminuir ou permanecer inalterada.

Assuma agora que NET é menor que o limite. Então δ OUT_j só pode ser negativo ou zero; daí, novamente a energia deve ou diminuir ou permanecer constante.

Finalmente, se NET é igual ao limite, δ_j é zero e a energia permanece inalterada.

2.6.5 - Memória Associativa

A memória humana opera de maneira associativa; isto é, uma porção de coleções pode produzir uma memória relacionada. Já a memória dos computadores é uma localização endereçável onde está contido o dado.

Uma rede recorrente forma uma memória associativa. Como na memória humana , uma porção dos dados desejados é fornecida e a "memória" completa é obtida. Para fazer uma memória associativa usando uma rede recorrente, os pesos devem ser selecionados para produzir energia mínima.

Hopfield desenvolveu uma memória associativa na qual as saídas são contínuas, na faixa de +1 a -1, correspondendo aos valores binários 0 e 1, respectivamente. As memórias são codificadas como vetores binários e armazenadas nos pesos de acordo com a seguinte fórmula:

$$w_{ij} = \sum_{d=1}^{m}\left(OUT_{id}OUT_{jd}\right) \tag{2.31}$$

onde

m = o número de memórias desejadas (vetores de saída)

d = o número de memórias desejadas (vetor de saída)

OUT = a i-ésima componente do vetor de saída desejado

Esta expressão pode ser simplificada notando que cada matriz de pesos **W** pode ser achada calculando o produto de cada vetor desejado com ele mesmo transposto (se o vetor desejado tem n componentes, esta operação forma uma matriz nxn) e em seguida somando todas as matrizes. Isto pode ser expresso simbolicamente como segue:

$$W = \sum_j D_j^t D_j \tag{2.32}$$

onde $\mathbf{D}_j$ é o j-ésimo vetor linha desejado.

Uma vez que os pesos são determinados, a rede pode ser usada para produzir o vetor de saída desejado, mesmo que o vetor de entrada seja parcialmente incorreto ou incompleto. Então, as saídas da rede são primeiramente forçadas para valores deste vetor de entrada. Em seguida, o vetor de entrada é removido e a rede é permitida "relaxar". Com isso a rede segue uma inclinação local da função de energia, e pode ficar em um mínimo local, não achando a melhor solução.

2.6.6 - Sistemas Contínuos

Hopfield mostrou outros casos nos quais a função de ativação **F** é contínua, obtendo com isso mais precisão na simulação do neurônio biológico. Uma escolha comum é a função sigmoidal em forma de S ou função logística.

$$F(x) = 1/\left(1+e^{-\lambda NET}\right) \tag{2.33}$$

onde λ é o coeficiente que determina o tamanho do passo da função sigmoide. Se λ é grande, F aproxima-se da função limite descrita anteriormente; pequenos valores para λ produzem uma inclinação mais suave.

Como no sistema binário, a estabilidade é garantida se os pesos são simétricos.

2.7 - MEMÓRIA ASSOCIATIVA BIDIRECIONAL (BAM)

2.7.1 - Introdução

A memória humana é freqüentemente associativa, uma coisa nos lembra de outra e esta, por sua vez, nos lembra de ainda outra.

Uma memória associativa é chamada de auto-associativa quando ela pode ser completada ou corrigida, mas não pode ser associada com uma memória diferente. Isto é resultado de sua estrutura de uma única camada, a qual requer que o vetor de saída apareça nos mesmo neurônios aos quais o vetor de entrada foi aplicado.

A memória associativa bidirecional (BAM) é heteroassociativa; isto é, ela pode aceitar um vetor de entrada em um conjunto de neurônios e produzir um relacionado, mas diferente, vetor de saída em outro conjunto de neurônios. Por exemplo, se é colocado na entrada da rede o padrão "1" , a saída de rede pode ser "um " ao invés de "1". Como a rede Hopfield, a BAM é capaz de generalização, produzindo saídas corretas apesar de entradas corrompidas. Também versões adaptativas podem ser abstratas, extraindo o ideal de um conjunto de exemplos com ruído. Essas características assemelham-se fortemente à certas funções do cérebro humano e colocam as redes neurais artificiais (ANN) um passo mais próximo da emulação de cérebro humano.

Desenvolvida por Bart Kosko, BAM é uma rede neural baseada em memória endereçada por conteúdo (content-addressable memories) [6]. BAM é globalmente estável e providência lembrança instantânea de qualquer de dois pares padrões. Contudo, BAM tem algumas limitações. Para grandes quantidades de padrões (n), a exigência de memória cresce conforme n^2.

BAM foi desenvolvida para solucionar problemas de reconhecimento de padrões. O ser humano faz reconhecimento de padrões sem pensar como isso é feito. Por muitos anos tem-se "aprendido" como associar padrões com representações verbais ou escritas. A rede BAM é uma

tentativa de descrever como o cérebro humano realiza este processo de associação. Apesar dos ruídos nos dados apresentados para o cérebro, ele consegue associar o que um objeto ou padrão deve ser. Portanto, BAM é uma generalização da rede Hopfield. Ao invés de criar a matriz de pesos com o produto de um padrão com ele mesmo (auto associativa), pares de padrões são usados (associação de pares). Depois da construção da matriz de pesos, um dos dois padrões pode ser aplicado como entrada para se obter como saída o outro padrão do par.

- Estrutura da BAM

A figura 2.12 mostra a configuração básica de uma BAM. Este formato é consideravelmente diferente daquele usado por Kosko. Ele foi escolhido para realçar a similaridade com a rede Hopfield e para permitir extensões para sistemas com mais camadas. Aqui um vetor de entrada **A** é aplicado para os pesos da rede **W** e produz um vetor de neurônios de saída **B**. O vetor **B** é então aplicado para a transposta da primeira rede de pesos $\mathbf{W}^t$ a qual produz uma nova saída para o vetor A.

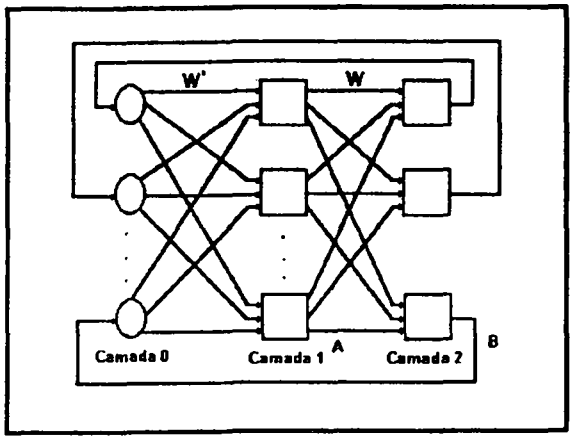

FIGURA 2.12 - Configuração de uma BAM

Este processo é repetido até a rede alcançar um ponto estável, quando nem **A** e nem **B** esta mudando. Note que os neurônios nas camadas 1 e 2 operam com outro paradigmas, produzindo a soma ponderada das entradas e aplicando-as para a função de ativação F. Este processo pode ser expresso em símbolos como segue:

$$b_i = F(\sum_j a_j w_{ij}) \qquad (2.34)$$

ou em forma vetorial:

$$B = F(AW) \tag{2.35}$$

onde:

B = vetor de saída da camada 2

A = vetor de saída da camada 1

W = matriz de pesos entre as camadas 1 e 2.

F = função de ativação

Similarmente:

$$A = F(BW^t) \tag{2.36}$$

onde W^t é a transposta da matriz W.

2.7.2 - Recuperando Uma Associação Armazenada

Memórias de longo prazo (***long-term***) (ou associações) são armazenadas nas matrizes W e W^t. Cada memória consiste de 2 vetores: **A**, o qual aparece na saída da camada 1, e **B**, a memória associada que é saída da camada 2. Para recuperar uma memória associada, todo ou parte do vetor **A** é momentaneamente colocado na saída da camada 1. **A** é então removido e a rede é permitida estabilizar, produzindo o vetor associado **B** na saída da camada 2. **B** então opera através da matriz transposta W^t para produzir uma réplica idêntica do vetor original de entrada **A** na saída da camada 1. Cada passo deste ***loop*** faz os vetores de saída das camadas 1 e 2 chegarem mais próximos da memória armazenada, até que um ponto estável é alcançado e as mudanças cessam. Este ponto pode ser considerado como um ponto de ressonância, quando os vetores passam para traz e para frente entre as camadas, sempre reforçando a saída corrente, mas sem sofrer grandes mudanças. O estado do neurônio representa uma memória curto prazo (***short-term***) (STM), porque ele pode mudar rapidamente pela aplicação de outro vetor de entrada. Os valores na matriz de pesos formam uma memória ***long-term***(LTM) e são mudados somente em grandes escalas de tempo, usando técnicas que serão mostradas.

Kosko mostrou que a rede esta atuando para minimizar uma função de energia Liapunov da mesma maneira que a rede Hopfield converge. Portanto, cada passo faz com que o sistema fique mais próximo de um mínimo de energia, cuja a localização é determinada pelo valores dos pesos.

2.7.3 - Codificando as Associações

Uma rede é usualmente treinada para reconhecer um conjunto de memórias. Este compreende um conjunto de treinamento, composto de pares de vetores **A** e **B**. A matriz de pesos é computada para ser a soma dos produtos de saída de todos os pares no conjunto de treinamento. Simbolicamente:

$$W = \sum_i A_i^t B_i \tag{2.37}$$

Kosko tem alcançado melhor desempenho fazendo os vetores bipolares antes de efetuar o produto de saída. Portanto, um componente do vetor maior que zero será 1 e menor ou igual a zero será -1.

BAM tem a capacidade para generalizar. Por exemplo, se um vetor **A** incompleto ou parcialmente incorreto é aplicado, a rede tende a produzir a memória mais próxima em **B**, o qual tende a corrigir o erro em **A**. Vários passos podem ser exigidos, mas a rede converge para a memória armazenada mais próxima.

Sistemas realimentados podem ser propensos à oscilações; eles podem oscilar de um estado a outro e nunca estabilizar. Kosko provou, contudo, que todas as BAM's são incondicionalmente estáveis para qualquer matriz de pesos. Esta importante característica vem do relacionamento transposto entre as duas matrizes de pesos, e significa que qualquer conjunto de associações pode ser apreendido sem risco de instabilidade.

2.7.4 - Capacidade da Memória

Como a rede Hopfield, BAM tem restrições no número máximo de associações que ela pode precisamente lembrar. Se este limite é excedido, a rede pode produzir saídas incorretas, lembrando associações que não foram ensinadas.

Algumas estimativas dizem que com 1024 neurônios na menor camada pode-se lembrar cerca de 25 associações. Este resultado desencorajador implica que grandes sistemas podem armazenar somente modestos números de associações.

Haines e Hecht-Nielsen (apud[1]) mostraram que BAM pode ter até 2^n estados estáveis, se um termo limite (*threshold*) **T** é selecionado para cada neurônio.

2.7.5 - BAM Adaptativa

Nas versões de BAM consideradas até aqui, a matriz de pesos é calculada como a soma do produto das saídas dos pares de vetores de entrada. Este cálculo é útil para demonstrar as funções que uma BAM pode representar. Contudo, certamente não é a maneira com que as conexões sinápticas são determinadas no cérebro.

A BAM adaptativa ajusta seus pesos durante a operação; isto é, a aplicação do conjunto de vetores de treinamento faz com que ela verifique sua própria energia. Gradualmente a STM transforma-se em LTM, modificando a rede como uma função de sua experiência. A rede é treinada pela aplicação dos vetores à camada **A** e os vetores associados a camada **B**. Se um ou ambos vetores são corrompidos dentro de um limite, a rede aprende a idealizar os vetores livres de ruídos. Neste sentido, ela abstrai a essência das associações, aprendendo padrões ideais apesar de ter visto somente padrões ruidosos.

2.7.6 - BAM Competitiva

Em muitos sistemas biológicos neurais, algum tipo de competição é observada entre neurônios.

Na BAM, a competição é implementada pela interconexão dos neurônios dentro de cada camada por meio de pesos adicionais. Isto forma outra matriz de pesos, com pesos positivos na diagonal principal e negativos em outras posições. O teorema de Cohen-Grossberg (apud [1]) mostra que tais redes são simétricas. Contudo, não é conhecida qual classe de pesos poderia levar a uma operação não estável.

2.7.7 - Discussão

Os dois grandes problemas de BAM são a limitada capacidade de armazenamento e a dificuldade para lembrar quando o número de padrões armazenados excede sua capacidade.

A rede BAM, no entanto, tem muitas vantagens. Ela é compatível com circuitos analógicos e sistemas óticos, e converge rapidamente tanto na fase de aprendizado quanto na fase de teste.

CAPÍTULO III

APRESENTAÇÃO DE ALGUNS MÉTODOS BÁSICOS DE PROCESSAMENTO DE IMAGEM

3.1 - Introdução

Somente nesta última década o processamento de imagem tornou-se significante em aplicações industriais. Uma das principais razões para isso é econômica; como qualquer outra aplicação industrial, é necessário mostrar os benefícios econômicos obtidos com a sua adoção. Talvez por essa razão o desenvolvimento tenha ocorrido primeiro na área militar (reconhecimento aéreo), depois em áreas não militares (sensoriamento remoto), medicina (tomografia) e no comércio (reconhecimento de caracteres). Em todas essas áreas, com exceção da área comercial, a viabilidade econômica é deixada um pouco em segundo plano e na área comercial a perspectiva de benefícios financeiros impulsiona o desenvolvimento.

As aplicações industriais de processamento de imagem podem ser basicamente divididas em dois grupos: aquele onde o processamento de imagem por computador substitui a visão humana e aquele onde ele substitui outro método de processamento de imagem. Isto é conhecido formalmente como *Machine vision* ou *Computer vision*, e envolve tarefas que eram anteriormente desempenhadas (ou no mínimo poderiam ter sido desempenhadas) por humanos.

Visão por computador pode ser dividida em quatro categorias: (1) visão de robô, (2) inspeção, (3) conversão de linha-desenhada e (4) modelamento de objetos, das quais as duas primeiras são de importância especial [10]. Machine vision recebe e interpreta uma imagem de uma cena real para obter informações e/ou controlar máquinas ou processos. Portanto, machine vision envolve interpretação automática de imagens para controle de processos, controle de qualidade, controle de máquina e controle de robô. Um típico sistema machine vision consiste de

uma câmera eletrônica, um computador e uma interface que funciona como pré-processador, o qual irá digitalizar a imagem obtida pela câmera e processá-la no computador [9].

Existem muitas técnicas convencionais de processamento de imagem utilizadas em automação industrial. Essas técnicas envolvem análise de histograma, detecção de margens etc. A escolha de uma dessas técnicas é baseada no tipo de aplicação. Analisaremos a seguir algumas destas técnicas.

3.2 - Histograma de Nível de Cinza

Uma câmera eletrônica usualmente representa a imagem por meio de uma matriz de pontos ou "pixels". Cada pixel, em uma câmera preto-e-branco, é caracterizado por um "nível de cinza" (ou intensidade de luz naquele ponto), representado digitalmente por um byte e podendo assumir valores entre 0 e 255 [11].

Uma das mais simples e úteis ferramentas no processamento de imagem digital é o histograma de nível de cinza de uma imagem. Esta função sumariza o nível de cinza de uma imagem. O histograma de qualquer imagem contém considerável informação e muitos tipos de imagem são completamente especificados pelo histograma.

O histograma de níveis de cinza é uma função mostrando, para cada nível de cinza, o número de pixels na imagem que têm aquele nível de cinza. A abscissa é o nível de cinza e a ordenada é o número de ocorrências (número de pixels). A figura 3.1 mostra uma peça e a figura 3.2 mostra o seu histograma.

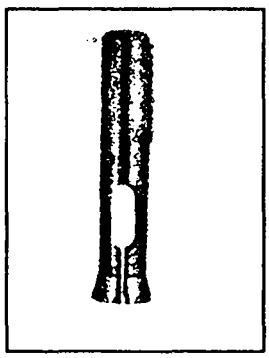

FIGURA 3.1 - Exemplo de uma peça

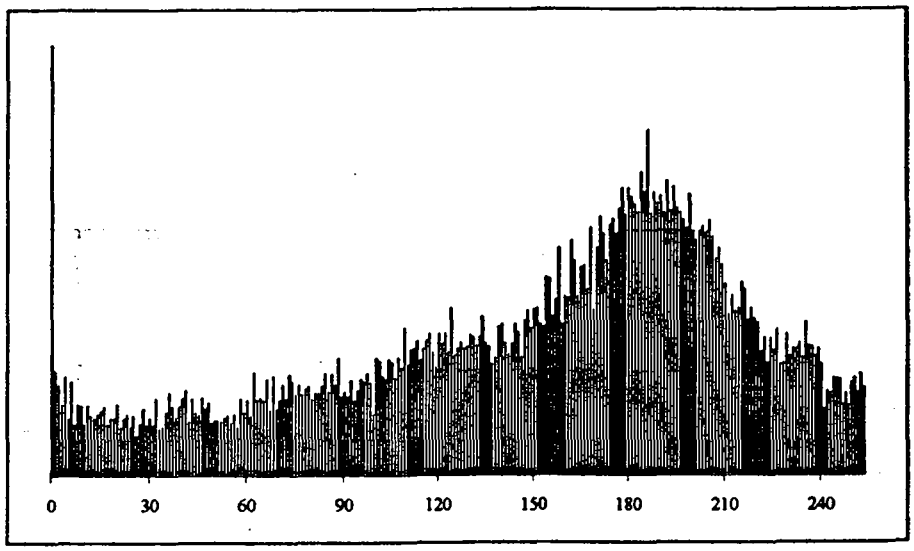

FIGURA 3.2 - Histograma da peça exemplo

Existe uma outra maneira para definir o histograma de nível de cinza. Supondo uma imagem contínua, definida pela função D(x,y), a qual varia lentamente de um nível de cinza alto no centro para um nível de cinza baixo nas bordas, pode-se selecionar algum nível de cinza D_i e definir um conjunto de linhas conectando todos os pontos da imagem tendo o nível de cinza D_1. As linhas de contorno resultantes formam uma curva fechada delimitando uma área que possui níveis de cinza iguais ou maiores que D_1. A figura 3.3 ilustra este princípio, ela mostra uma imagem contendo uma linha de contorno no nível de cinza D_1. Uma segunda linha de contorno foi desenhada em um nível de cinza mais alto D_2.

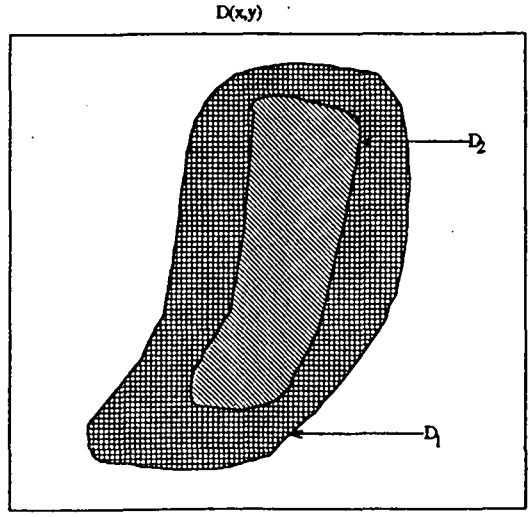

FIGURA 3.3 - Linhas de contorno de uma imagem

Quando uma imagem é condensada em um histograma, toda informação espacial é descartada. O histograma especifica o número de pixels tendo cada nível de cinza, mas não diz onde estes pixels são localizados na imagem. Portanto, o histograma é único para qualquer imagem, mas o inverso não é verdade. Imagens diferentes podem ter histogramas idênticos. Operações como mover um objeto dentro de uma imagem, poderiam não produzir efeito no histograma. Entretanto, tem-se que observar as dificuldades para se produzir uma iluminação homogênea de modo que, apesar da rotação e translação, os mesmos pixels não mudem o tom de cinza.

3.3 - Ponto de Operação

Ponto de operação constitui uma simples, mas importante classe das técnicas de processamento de imagem. Ela permite ao usuário modificar a maneira com a qual seus dados enchem a área disponível de níveis de cinza. Um ponto de operação é, por definição, uma operação que pega uma imagem na entrada e transforma em outra imagem, de maneira que o nível de cinza de cada pixel na saída depende unicamente do nível de cinza do pixel da entrada correspondente. Além disso, em um ponto de operação, cada pixel de saída corresponde diretamente ao pixel de entrada tendo a mesma coordenada. Portanto um ponto de operação não pode modificar o relacionamento espacial dentro de uma imagem. Pontos de operação são algumas vezes chamados por outros nomes, como "intensificação de contraste" (*constrast enhancement)* e "manipulação da escala de cinza" [11].

Ponto de operação modifica a escala de cinza de uma imagem. Isto pode ser visto como uma operação de cópia pixel-por-pixel, exceto que os níveis de cinza são modificados de uma maneira pré-determinada. Um ponto de operação que transforma uma imagem de entrada A(x,y) em uma imagem de saída B(x,y) pode ser expresso matematicamente como:

$$B(x,y) = f[A(x,y)] \tag{3.01}$$

O ponto de operação é completamente especificado pela função **f**, a qual especifica o mapeamento do nível de cinza da entrada para o nível de cinza da saída.

O método de ponto de operação pode ser usado para remover um efeito não linear de um sensor de imagem. Por exemplo, suponha que uma imagem tenha sido digitalizada por um instrumento com uma resposta não linear para uma determinada intensidade de luz. O ponto de operação pode transformar o nível de cinza, tal que os níveis de cinza representem incrementos iguais nesta intensidade de luz.

No caso anterior, o ponto de operação foi usado para melhorar a imagem antes de processá-la. Igualmente importante é o uso de ponto de operação antes de colocar a imagem num monitor. Muitos monitores não mantêm uma relação linear entre o nível de cinza de um pixel na imagem digital e o brilho deste pixel no monitor.

3.4 - Operações Algébricas

Operações algébricas são todas aquelas que produzem uma imagem de saída que é a soma, diferença, produto ou quociente pixel-por-pixel de duas imagens. No caso de soma, mais de duas imagens podem ser usadas. Adição, subtração, multiplicação e divisão por uma constante pode ser tratado como um ponto de operação, como mostrado anteriormente [11].

Uma importante aplicação da adição de imagens é quando se tem várias imagens da mesma cena. Isto é frequentemente útil para reduzir o efeito de um ruído aleatório aditivo. A subtração pode ser usada para remover um indesejável padrão aditivo de uma imagem. Subtração também é útil para detectar mudanças em duas imagem da mesma cena. Por exemplo, poderia-se detectar movimento pela subtração de imagens sequenciais da mesma cena. A multiplicação e a divisão possuem pouca aplicação em processamento digital de imagem. Elas podem ser usadas para corrigir os efeitos de um digitalizador no qual a sensibilidade do sensor de luz varia de um ponto a outro dentro da imagem.

3.5 - Operações Geométricas

Operações geométricas são aquelas que mudam o relacionamento espacial entre objetos dentro de uma imagem [11]. Essas operações podem ser imaginadas como movendo

alguma coisa dentro de uma imagem. Deste modo, qualquer ponto podem ser movido para qualquer lugar, mas deve-se preservar uma coerência neste movimento, pois, de outro modo, uma imagem somente seria um amontoado de pixels sem nenhum significado.

Em processamento digital de imagem usando operações geométricas deve existir um algoritmo que defina a transformação espacial. Isto especifica a "movimentação" de cada pixel, como ele move-se da sua posição inicial para sua posição final. Em muitas aplicações é desejável preservar as características e a conectividade de objetos na imagem. A definição geral para a operação geométrica é:

$$g(x,y) = f(x',y') = f[a(x,y),b(x,y)] \tag{3.02}$$

onde f(x,y) é a imagem de entrada e g(x,y) é a imagem de saída. As funções a(x,y) e b(x,y) especificam somente a transformação espacial.

Uma outra exigência para uma operação geométrica é um algoritmo para interpolação de valores de nível de cinza.

De posse da transformação espacial e um algoritmo para interpolação de nível de cinza, pode-se efetuar a operação geométrica. Geralmente, o algoritmo para interpolação é permanentemente estabelecido no programa de computador. O algoritmo definindo a transformação espacial, contudo, é especificado para cada tarefa. Desde de que o algoritmo de interpolação ou é o mesmo ou é um de lista de opções, é a transformação espacial que serve na prática para definir cada operação geométrica.

3.6 - Curvas de Distribuição de Intensidade

Este método assemelha-se bastante ao de histograma de nível de cinza. Consiste basicamente em somar todos os pixels dispostos como linha e coluna, ou seja, uma imagem digital contendo 512 x 512 pixels, seria interpretada com contendo 512 linhas e 512 colunas.

Os pixels dispostos em cada linha seriam somados para produzir uma curva que mostra o perfil do objeto contido na imagem. Do mesmo modo, os pixels em cada coluna seriam somados. Estas duas curvas caracterizam um objeto na imagem.

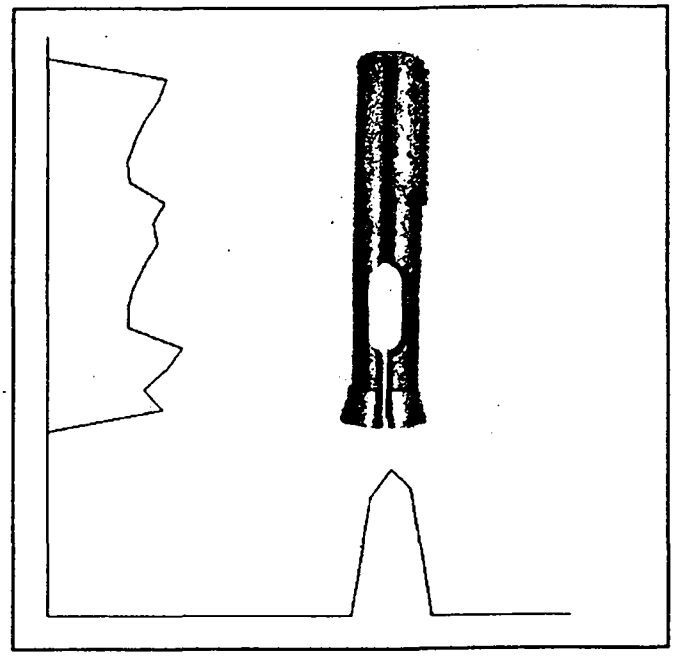

FIGURA 3.3 - Exemplo de curva de distribuição de intensidade

O mesmo objeto deslocado de sua posição poderia ser representado pelas mesmas curvas de intensidade e considerando que objetos distintos não podem ser descritos pela mesma imagem, pode-se usar este método para pré-processar uma imagem, extraindo características importantes de cada objeto.

Este método de pré-processamento de imagem foi um dos adotados neste trabalho. Sua implementação é simples, mas possui algumas restrições. Por exemplo, somente um objeto deve-se encontrar na cena para que seja completamente representado pelas curvas. A figura 3.3 mostra a curva de distribuição de intensidade de uma determinada peça.

3.7 - Detecção de Objetos numa Imagem

A detecção de objetos numa imagem esta relacionada com a presença ou ausência de objetos suspeitos de estarem presente nesta imagem.

3.7.1 - Igualando Modelos

Um dos mais fundamentais meios de detectar um objeto numa imagem é iguala-lo com um modelo, que é uma replica do objeto de interesse, esse modelo é comparado com todos os objetos desconhecidos contidos na imagem esperando que um deles iguale-se ao modelo.

Como um simples exemplo deste processo considere um conjunto de figuras geométricas como mostrado na figura 3.4-a. Neste exemplo o objetivo é detectar a presença e a localização de um triângulo na imagem [12]. A figura 3.4-b mostra o modelo deste triângulo.

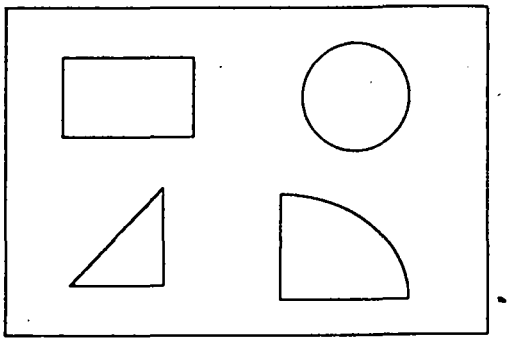

FIGURA 3.4-a - Imagem com objetos

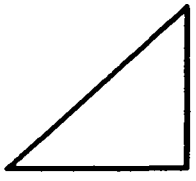

FIGURA 3.4-b - Modelo do triângulo

A largura das linhas que formam o triângulo modelo é escolhida como um compromisso entre a localização precisa e o tamanho invariante do modelo. Na operação de detecção o modelo é passado sobre a imagem e a região comum entre o modelo e a imagem é comparada.

Um modelo é raramente exatamente igual ao objeto na imagem, devido a ruído na imagem, efeito de quantização da amplitude e uma incerteza a priori de como a forma exata e a estrutura do objeto será detectada. Consequentemente, um procedimento comum é produzir a medida da diferença D(m,n) entre o modelo e o campo da imagem em todos os pontos do campo

da imagem onde $-M \leq m \leq M$ e $-N \leq n \leq N$. Um objeto é dito igualar um modelo onde quer que a diferença é menor que algum nível estabelecido $L_D(m,n)$. Normalmente o limite é constante para cada imagem. A medida de diferença geralmente utilizada é a diferença dos mínimos quadrados ou erro definida por :

$$D(m,n) = \sum_j \sum_k [F(j,k) - T(j-m,k-n)]^2 \tag{3.03}$$

onde F(j,k) denota o campo da imagem a ser procurado e T(j,k) é o modelo. A procura é restrita para região sobreposta entre o modelo e o campo da imagem. Diz-se então que existe um igualamento na coordenada (m,n) se

$$D(m,n) < L_D(m,n) \tag{3.04}$$

Expandindo a equação (3.03) tem-se:

$$D(m,n) = D_1(m,n) - 2D_2(m,n) + D_3(m,n) \tag{3.05}$$

onde:

$$D_1(m,n) = \sum_j \sum_k [F(j,k)]^2 \tag{3.06}$$

$$D_2(m,n) = \sum_j \sum_k F(j,k)T(j-m,k-m) \tag{3.07}$$

$$D_3(m,n) = \sum_j \sum_k [T(j-m,k-n)]^2 \tag{3.08}$$

O termo $D_3(m,n)$ representa a somatória da energia do modelo e é um valor constante independente do coordenada (m,n). A energia da área da imagem em análise representada pelo termo $D_1(m,n)$ geralmente varia lentamente sobre a imagem. O termo $D_2(m,n)$ deve ser reconhecido com sendo a correlação cruzada $R_{FT}(m,n)$ entre o campo da imagem e modelo. Na localização da coordenada do modelo, a correlação cruzada deve ser

grande para dar uma pequena diferença. Contudo, a magnitude da correlação cruzada não é sempre uma medida adequada da diferença do modelo porque o termo da energia da imagem $D_1(m,n)$ é invariante da posição. Por exemplo, a correlação cruzada pode ser grande mesmo numa situação de não igualamento, se a amplitude da imagem sob a região do modelo é maior sobre uma coordenada particular (m,n). Esta dificuldade pode ser evitada pela comparação da correlação cruzada normalizada.

$$R'_{FT}(m,n) = \frac{D_2(m,n)}{D_1(m,n)} = \frac{\sum_j \sum_k F(j,k)T(j-m,k-m)}{\sum_j \sum_k [F(j,k)]^2} \tag{3.09}$$

com um nível limite $L_R(m,n)$. A igualdade acontece se:

$$R'_{FT}(m,n) > L_R(m,n)$$

O valor máximo unitário da correlação cruzada ocorre se e somente se a função de imagem sob o modelo iguala exatamente o modelo.

Uma das maiores limitações deste método é que um enorme número de modelos deve ser utilizado para levar em conta a rotação, translação e magnitude do objeto na imagem. Por esta razão, este método é geralmente usado para casos particulares, nos quais o tamanho e a forma do objeto não variem muito.

3.8 - Técnicas de Processamento de imagem Utilizadas em Inspeção

Existem muitas técnicas de processamento de imagem úteis para tarefas de inspeção. Uma breve descrição de duas delas será dada aqui [9]. Ambas só podem ser usadas em imagens binárias, as quais contém somente pixels preto e branco.

- Cálculo de Área (*Area Calculation*) - Nesta técnica , uma região da imagem é limitada. Este limite é representado por aqueles pixels que possuem pixels vizinhos tanto pretos quanto brancos. Na aplicação desta técnica é calculado o número de pixels pretos que estão

contidos na região selecionada e no seu limite. Um valor limite para inspeção é predeterminado e implementado em algoritmo para distinguir a peça boa da peça defeituosa. A área de todas as peças boas deve ser bastante próxima para todos os objetos e sempre maior que a de uma peça defeituosa.

- Determinação do Número de Pontos do Limite (***Boundary Following***) - O primeiro passo desta técnica é detectar o limite por um procedimento proposto por Papert (apud [9]). O limite é detectado no sentido horário e o número de pontos é armazenado durante o traçado. O número de pontos do limite da peça de teste é então comparado com os pontos da peça boa. Baseado na diferença entre o número dos pontos do limite, é feita a distinção entre a peça boa e a peça defeituosa. Quando a diferença esta dentro de uma variação predeterminada, a peça é considerada boa.

3.9 - Processamento de Imagem e Visão de Robô

Processamento de imagem envolve todas as técnicas de computador que usam imagens como dados de entrada. Essas técnicas podem ser muito diferentes, para as diferentes áreas que as usam.

Processamento de imagem e visão de robô envolvem ambas tratamento de uma cena com um computador, mas suas exigências básicas são distintas.

As principais exigências para um bom sistemas de visão de robô são [10]:

- Confiabilidade de operação;

- Simplicidade no ajuste e programação;

- Rápido processamento de imagem (geralmente em tempo real);

- Fácil iluminação da cena;

- Baixo custo.

Muitas vezes essas exigências são opostas a resultados oriundos de grupos trabalhando com processamento de imagem. Por exemplo, excelentes resultados podem ser obtidos trabalhando com níveis de cinza, como melhoramento de contraste, redução de ruído, compressão de dados etc. Mas estes resultados geralmente são obtidos com grandes computadores ou elevado tempo de processamento.

Provavelmente, um simples aspecto que pode melhorar um sistema de visão de robô é o controle da iluminação da área trabalhada. Este aspecto foi inicialmente negligenciado, mas posteriormente pesquisadores conseguiram várias técnicas que simplificaram grandemente o processamento de imagem (estrutura iluminada, feixe de laser etc). Algumas dessas técnicas são mostradas na figura 4.4. A desvantagem do controle da iluminação é a sua restrição para o caso específico para o qual a iluminação foi projetada. Bons resultados obtidos com uma iluminação especifica podem não ser úteis quando da ocorrência de mudanças na área de trabalho.

Idealmente, um sistema de visão de robô deve ser capaz de pegar uma cena, analisá-la e interpretá-la corretamente. Todas esses funções devem ser feitas em tempo real.

Neste sentido, a utilização de redes neurais vem mostrando que estas podem ser uma solução para estes sistemas. A partir do momento que a rede esta treinada (treinamento *off line*), basta colocar um padrão na sua entrada que a resposta é quase instantânea. Outras habilidades das redes neurais como capacidade para tratar com padrões corrompidos e capacidade de generalização também são importantes.

Se não bastasse essas características, supondo-se por exemplo que se utilize o método de igualamento de modelo, para cada variação de um objeto teria-se que ter um modelo armazenado para tentar igualá-lo com o objeto em questão. É fácil perceber a grande capacidade de armazenamento necessária para isto, sem falar no tempo de processamento.

Sistemas híbridos utilizando técnicas convencionais e redes neurais podem vir a ser também uma boa solução.

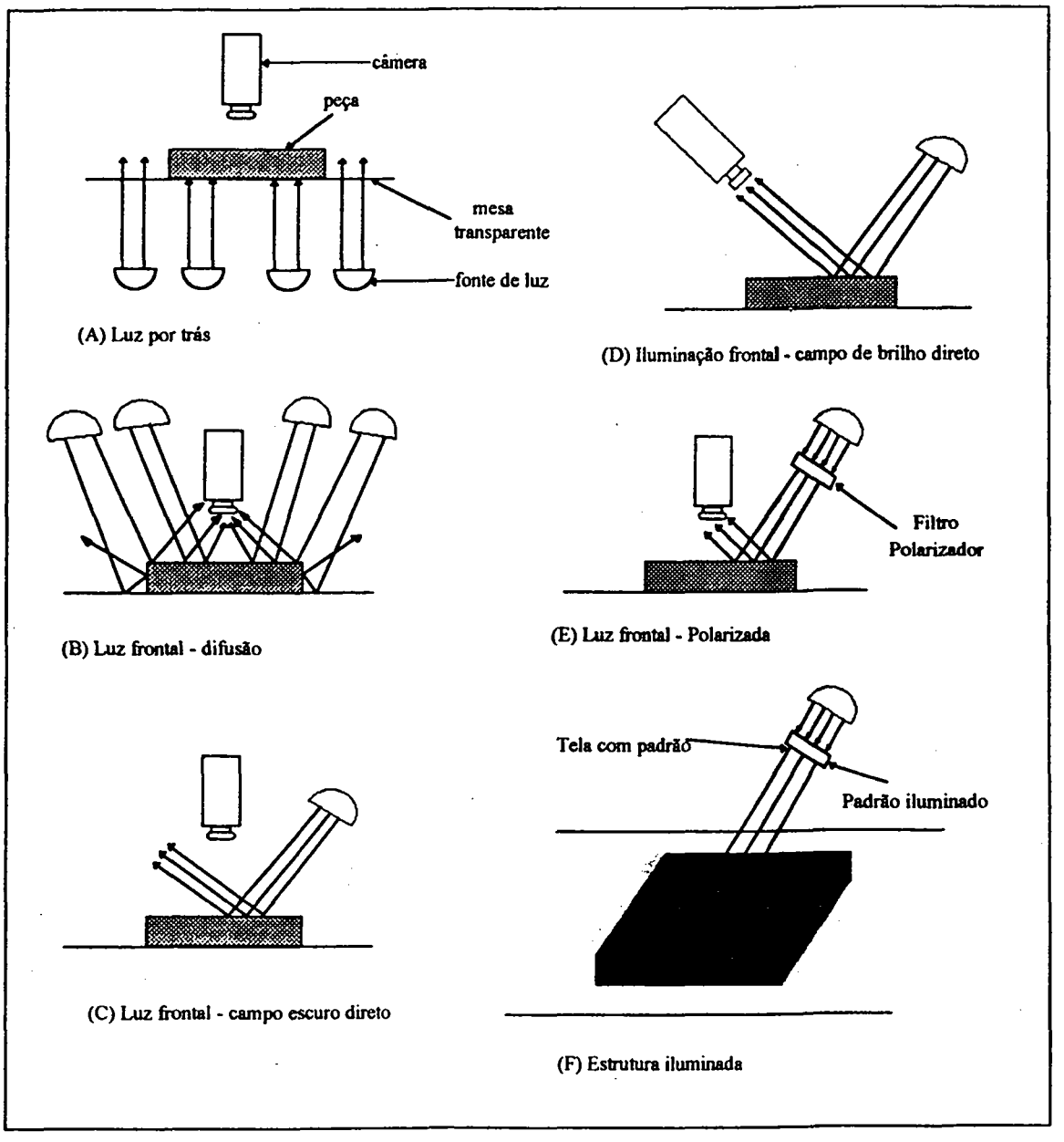

FIGURA 4.4 - Técnicas de iluminação

3.10 - Desempenho dos Sistemas Comerciais de Visão de Robô

Devido a ampla diferença entre os sistemas de visão de robô é difícil comparar seus desempenhos. Algumas poucas características dão uma idéia das possibilidades de variações que podem ser encontradas na indústria. Resolução, velocidade de processamento e confiabilidade são características freqüentemente empregadas para mostrar a capacidade desses sistemas. A primeira é uma característica absoluta. as duas últimas são dependentes da aplicação. Duas importantes

características freqüentemente não encontradas em sistemas comerciais de visão de robô são processamento 3D e reconhecimento de objetos parcialmente oclusos [10].

A percentagem de acerto depende muito do tipo de objeto utilizado, das condições da cena (por exemplo, iluminação), da quantidade de treinamento realizado no sistema e de outros fatores. É geralmente aceito como boa uma taxa de acerto de 95% em tarefas de controle de qualidade e inspeção [10].

Considerações especiais são dadas para aplicações em tempo real. Tempo real num ambiente de manufatura pode ser compreendido com o máximo tempo que o processo de manufatura permite ao sistema de visão robótica para tomar uma decisão antes que o movimento do robô seja efetuado. Outra interpretação pode ser derivada do fato de que algumas operações do robô tentam substituir um operador humano. Pesquisas mostram [10] que um homem leva entre 0.5 a 2 segundos para reagir quando ele estar usando seu sistema de visão para reconhecer características. Em aplicações industriais, para alcançar uma velocidade de processamento igual ou acima do que foi mencionado existem três alternativas. A primeira é o aumento da capacidade de processamento, adquirindo computadores mais potentes, a segunda é projetar chips VLSI incorporando muito do software em ROM, PLA ou outra tecnologia integrada e a terceira é a adoção de novos paradigmas de programação, como por exemplo, redes neurais.

CAPÍTULO IV

IMPLEMENTAÇÃO DE UM SISTEMA DE RECONHECIMENTO DE PEÇAS BASEADO EM REDES NEURAIS

4.1 - Introdução

Existe uma Célula Flexível de Manufatura (FMC - Flexible Manufacturing Cell), atualmente sendo operacionalizada em um trabalho conjunto do LCMI, GRUCON e fundação CERTI em uma área reservada para este fim no Departamento de Engenharia Mecânica (EMC). A estrutura pretendida da célula, já parcialmente implantada, é mostrada na figura 4.1. O sistema de transporte (esteira) e a câmera mostradas na figura ainda não estão disponíveis.

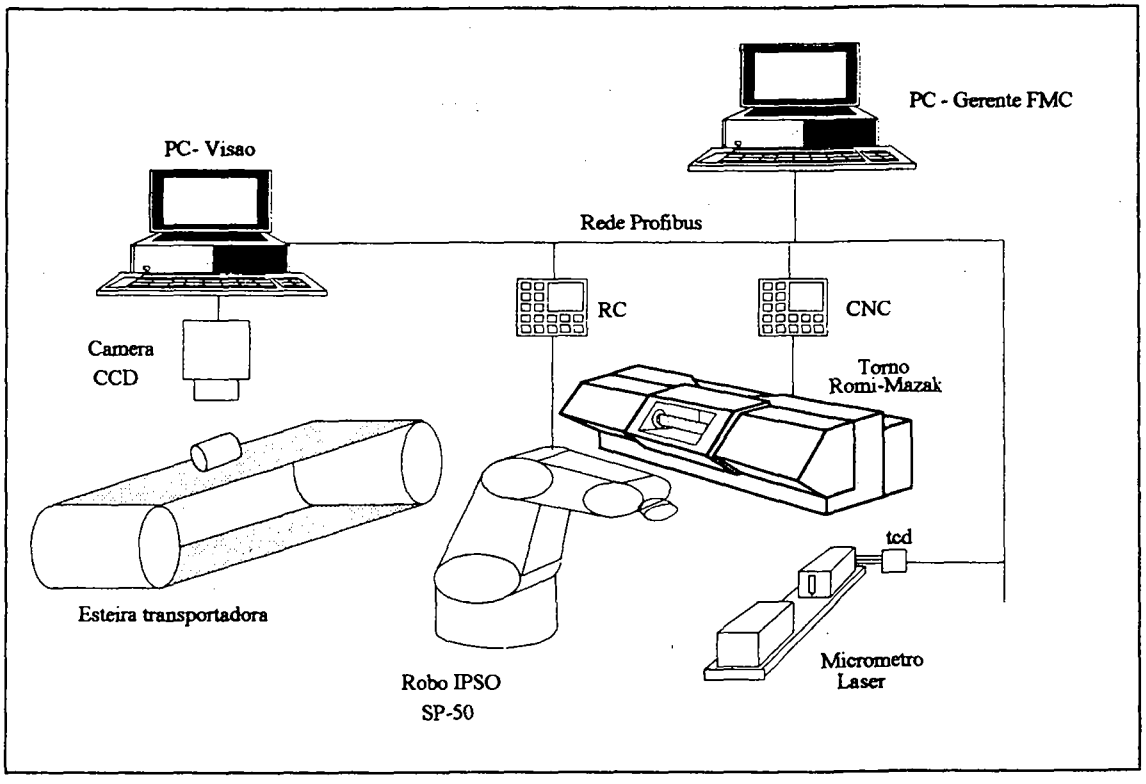

FIGURA 4.1 - Estrutura da FMC em implantação

Para fins de teste em uma primeira etapa, e independentemente da conclusão dos trabalhos de implantação desta FMC, o pacote de software de reconhecimento de peças será implementado inicialmente em uma Célula Flexível de Montagem miniaturizada, disponível no LCMI, que permite a realização de trabalhos de pesquisa nas áreas de automação da manufatura, informática industrial e controle de processos bem como de trabalhos didáticos ligados ao curso de graduação em Engenharia de Controle e Automação Industrial.

A estrutura miniaturizada disponível inclui um sistema de transporte de peças sob a forma de uma esteira rolante com sensores óticos que indicam a presença de peças ou pallets em posições chaves, três robôs industriais de pequeno porte integrados à célula, um sistema de visão baseado em uma câmera CCD (Charge Coupled Device) com placa de aquisição de imagem para computadores tipo PC. A estrutura é comandada por intermédio de um Controlador Lógico Programável (CLP). O algoritmo de reconhecimento deverá ser integrado à estrutura já existente. Para tal, um pacote de software adicional deverá ser desenvolvido, que inclui as funcionalidades de:

- leitura dos sensores óticos que indicam a presença de objetos na esteira em frente à câmera CCD;
- comando de parada a partida da esteira através de interação com o CLP;
- leitura e digitalização da imagem da câmera CCD;
- alimentação do algoritmo de reconhecimento neural com as imagens digitalizadas e conseqüente uso dos resultados obtidos para realizar um controle inteligente de comportamento do robô;
- transformação das coordenadas de posição e rotação da peça do referencial da câmera para o referencial do robô manipulador, de forma a permitir um correto posicionamento e orientação da garra;
- comando do robô para pegar (ou não) a peça;

- contagem do número de peças de cada tipo reconhecidas pela câmera.

Na figura 4.1 a estrutura composta pelo Gerente da FMC, o robô , o torno e o micrometro já esta implementada, nesta implementação o robô pega uma determinada peça em uma posição fixa, um operador determina o tipo de usinagem a ser feita e a peça então é usinada no torno, após a usinagem a peça é levada ao micrometro e classificada como "peça boa" ou "peça refugo".

Este trabalho pode ser visto na figura 4.1 como constituído do PC-Visão, da câmera CCD e da esteira transportadora. A imagem da peça é capturada pela câmera CCD, a seguir é então pré-processada e os pontos obtidos deste processamento são então colocados na entrada da rede neural (que neste momento já deve ter sido treinada), o resultado do reconhecimento é colocado na rede Profibus de modo que o gerente de manufatura possa determinar as ações adequadas.

4.2 - Escolha de uma Arquitetura de Rede e Algoritmo de Treinamento

Existem várias aplicações de reconhecimento de objetos utilizando uma abordagem convencional, contudo as redes neurais têm mostrado grande facilidade de readaptação (configuração) na ocorrência de uma mudança no conjunto de peças a serem reconhecidas. Além disso após o seu treinamento (que é feito *off-line*), o tempo de reconhecimento é bastante reduzido (o suficiente para serem feito alguns produtos de pesos por entradas). Aliada a estas características pode-se acrescentar mais uma: sua capacidade de poder responder corretamente mesmo a entradas que contenham ruídos ou sejam parcialmente incompletas. Entretanto, não se pode assegurar que um novo conjunto de peças será aprendido tão bem como o anterior. Neste caso, alguns ajustes nos coeficientes como os de taxa de aprendizado podem ser necessários.

Tendo em vista os objetivos propostos, foi escolhida a rede Counterpropagation, desenvolvida por Robert Hecht-Nielsen, para implementar o sistema de reconhecimento de peças. Em comparação com Backpropagation ela é de uso mais restrito, entretanto seu tempo de treinamento é bem menor. Esta característica pode ser importante, tendo em vista a tendência

atual de maior diversificação e diminuição do tamanho dos lotes de peças produzidas, requerendo assim aprendizado freqüente de novas imagens.

Foram feitas várias tentativas de se treinar uma rede utilizando o algoritmo backpropagation, entretanto o tempo de treinamento mostrou-se muito grande. Devido a dificuldade de convergência desta rede, algumas técnicas que melhoram este tempo de convergência foram utilizadas (elas são citadas no capítulo 2), mas o resultado ainda não foi satisfatório.

A rede perceptrons não foi escolhida porque não consegue representar padrões complexos em sua estrutura de uma camada. A rede Hopfield é auto-associativa, não sendo, portando, adequada para este trabalho. Já a rede BAM não pode ser utilizada por precisar de um número muito grande de neurônios para poder lembrar corretamente alguns poucos padrões. Por estes motivos foi feita a escolha da rede counterpropagation.

Para efeito de treinamento e testes, foram feitas inicialmente imagens de 13 peças rotativas de uma mesma família, já usinadas e disponíveis para este trabalho. Estas imagens foram tomadas rotacionado cada peça de aproximadamente 10°, perfazendo um total de 36 imagens de cada peça. Para isso utilizou-se uma câmera CCD (Charge Coupled Device) que gerava uma imagem de 512 x 512 pixels, tendo cada pixel o tamanho de 1 byte. A área útil desta imagem é de 512 x 480 pixels. Cada pixel da imagem contém um nível de cinza que varia entre 0 e 255.

Verificou-se nesta etapa que a iluminação feita sobre a peça é de grande importância. A iluminação é feita de forma a não gerar sombras. A estrutura para obtenção das imagens esta mostrada na figura 4.2 e as imagens digitalizadas das peças utilizadas são mostradas na figura 4.4.

Desde de que seria totalmente inviável mapear diretamente cada pixel da imagem para um neurônio de entrada da rede neural (seriam necessários 245.760 neurônios) existe a

necessidade de pré-processar a imagem, de forma a extrair desta somente as características básicas que permitem distinguir uma peça das demais.

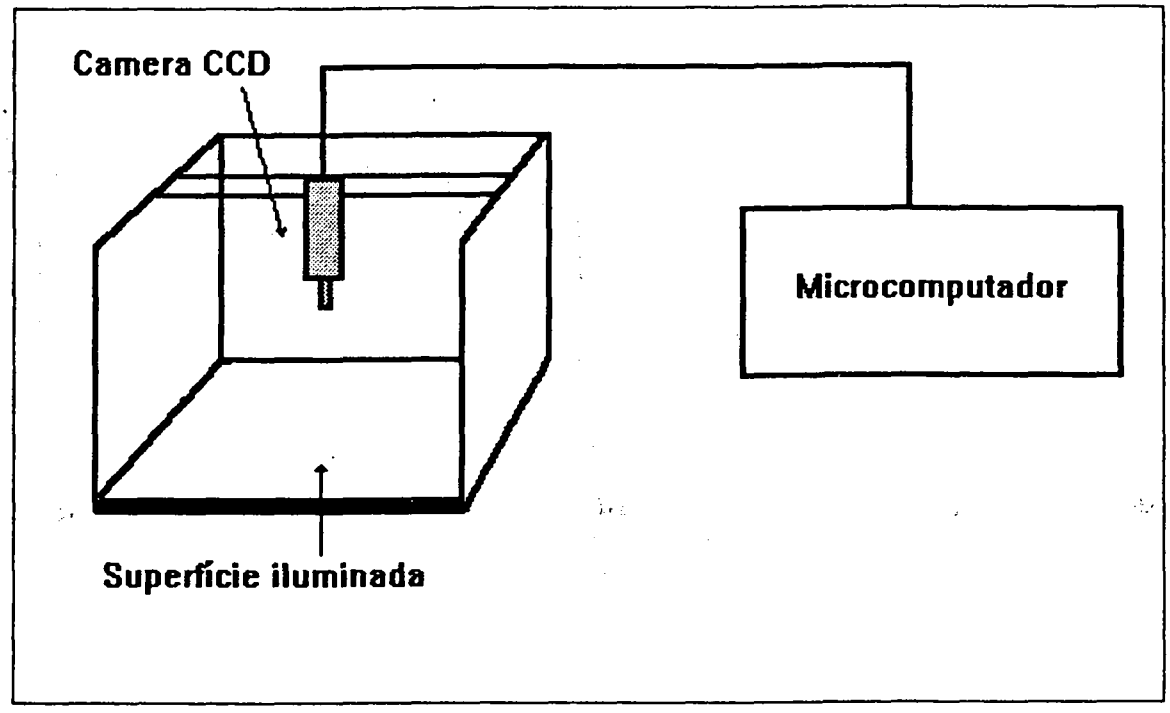

FIGURA 4.2 - Estrutura para obtenção das imagens

A primeira etapa para implementar esta rede neural foi gerar os pontos para sua entrada. Para tanto, por razões de simplicidade de implementação foi escolhido como método de pré-processamento da imagem o que traça curvas de distribuição de intensidade descrito no capítulo 3. Foi feito um programa de computador em Turbo C para ler a imagem, somar seus pixels em linhas e colunas, traçar as curvas e gerar os pontos para a rede neural. Numa etapa posterior foi implementado também o histograma de níveis de cinza, que possibilitou tecer algumas comparações.

A câmera usada foi uma COHU CCD monocromática modelo 4810 de alta resolução e alta sensibilidade.

A idéia principal por traz da escolha de um algoritmo simples de pré-processamento (curvas de distribuição de intensidade), foi verificar o quanto as redes neurais (counterpropagation especificamente), podem tratar com entradas corrompidas e/ou ligeiramente

diferentes do conjunto de entradas utilizado para treinamento. Ainda pelo motivo acima exposto, optou-se por não utilizar nenhum tipo de filtro na imagem. Com isso a tarefa de reconhecimento tornou-se mais rápida, mas também mais difícil. Os resultados deste trabalho poderá guiar este tipo de escolha em trabalhos posteriores. A utilização de um outro algoritmo de pré-processamento tão simples quanto o primeiro (histograma de níveis de cinza), mostrou o quanto o tipo de pré-processamento de imagem é importante para o aprendizado da rede neural. Com a utilização deste outro algoritmo pode-se ver que os erros de reconhecimento ocorreram em outros padrões, estando o erro portanto relacionado com o algoritmo de pré-processamento utilizado. Neste sentido, um algoritmo de pré-processamento que fosse independente da iluminação ou pelo menos retenha as características da peça apesar de mudanças na iluminação é grandemente desejável.

4.3 - Estrutura do Software do Sistema

O sistema de reconhecimento de peças aqui proposto compõem-se dos módulos:

- CURVA - gera os pontos para entrada da rede.

- RPNEURAL - treina e reconhece as peças utilizando os pontos gerados pelo módulo CURVA.

O software de leitura da imagem da câmera (SINTHE - Sistema Integrado de Holografia Eletrônica - Módulo de aquisição) utilizado para gerar os arquivos de imagem foi desenvolvido no LABMETRO (Laboratório de Metrologia) da Fundação CERTI (Centro Regional de Tecnologia e Informática de Santa Catarina) e foi aqui usado em cooperação com esta entidade, não tendo sido portanto desenvolvido neste trabalho. Este software captura a imagem vista pela câmera, e salva a mesma em arquivo. O padrão de salvamento de imagem é próprio da placa digitalizadora. Entretanto um conversor deste padrão (.IMG) para o padrão TIF encontra-se disponível com a placa digitalizadora.

A figura 4.3 mostra o diagrama de blocos esquemático do sistema.

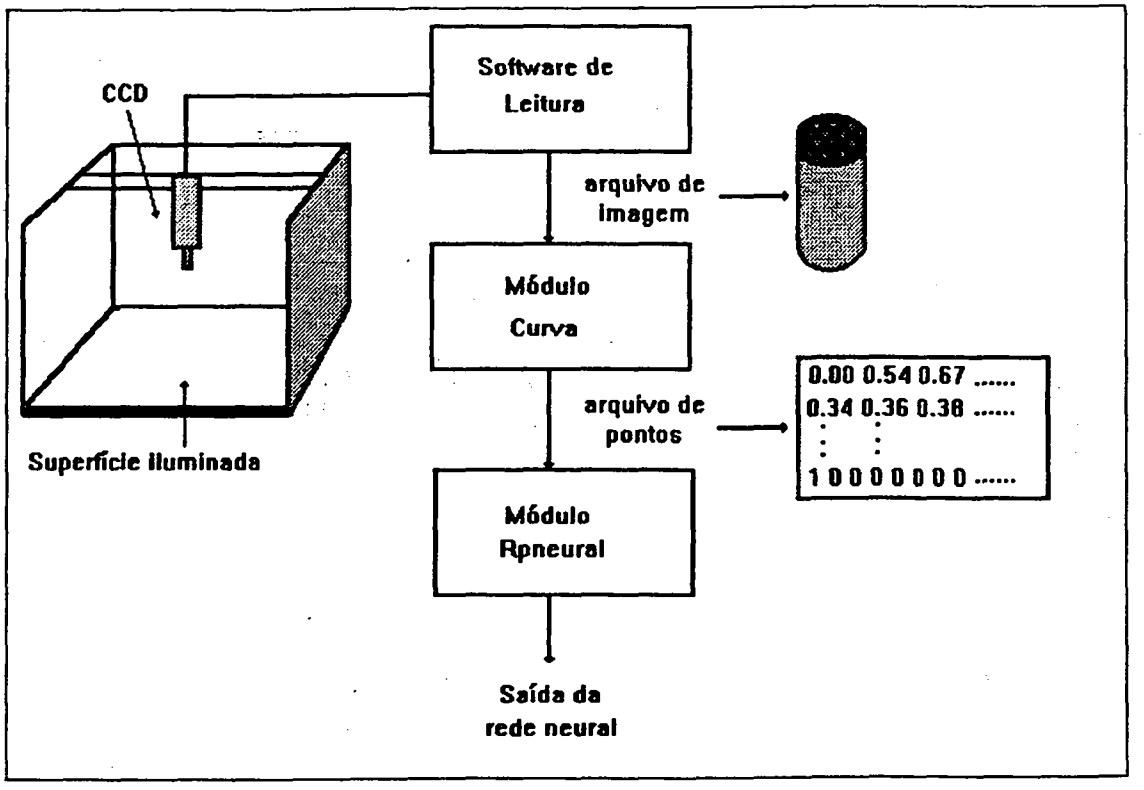

FIGURA 4.3 - Representação esquemática do sistema de reconhecimento de peças

4.3.1 - CURVA - Software para Traçar as Curvas de Distribuição de Intensidade e o Histograma de Níveis de Cinza.

Basicamente o objetivo deste módulo é gerar um arquivo de pontos para ser utilizado no treinamento da rede neural pelo módulo RPNEURAL. Este arquivo inclui no seu início uma descrição das características da rede. Isto deve ser feito na seguinte ordem: número de neurônios na camada de entrada, número de neurônios na camada Kohonen, número de neurônios Grossberg, valor do erro que permite dizer que a rede neural aprendeu (erro de parada), valor do parâmetro "alfa" (coeficiente de taxa de treinamento) e valor do parâmetro "beta" (coeficiente de proporcionalidade entre o neurônio e a saída desejada). As 13 peças utilizadas para efeito de aprendizado e testes são mostradas na figura 4.4.

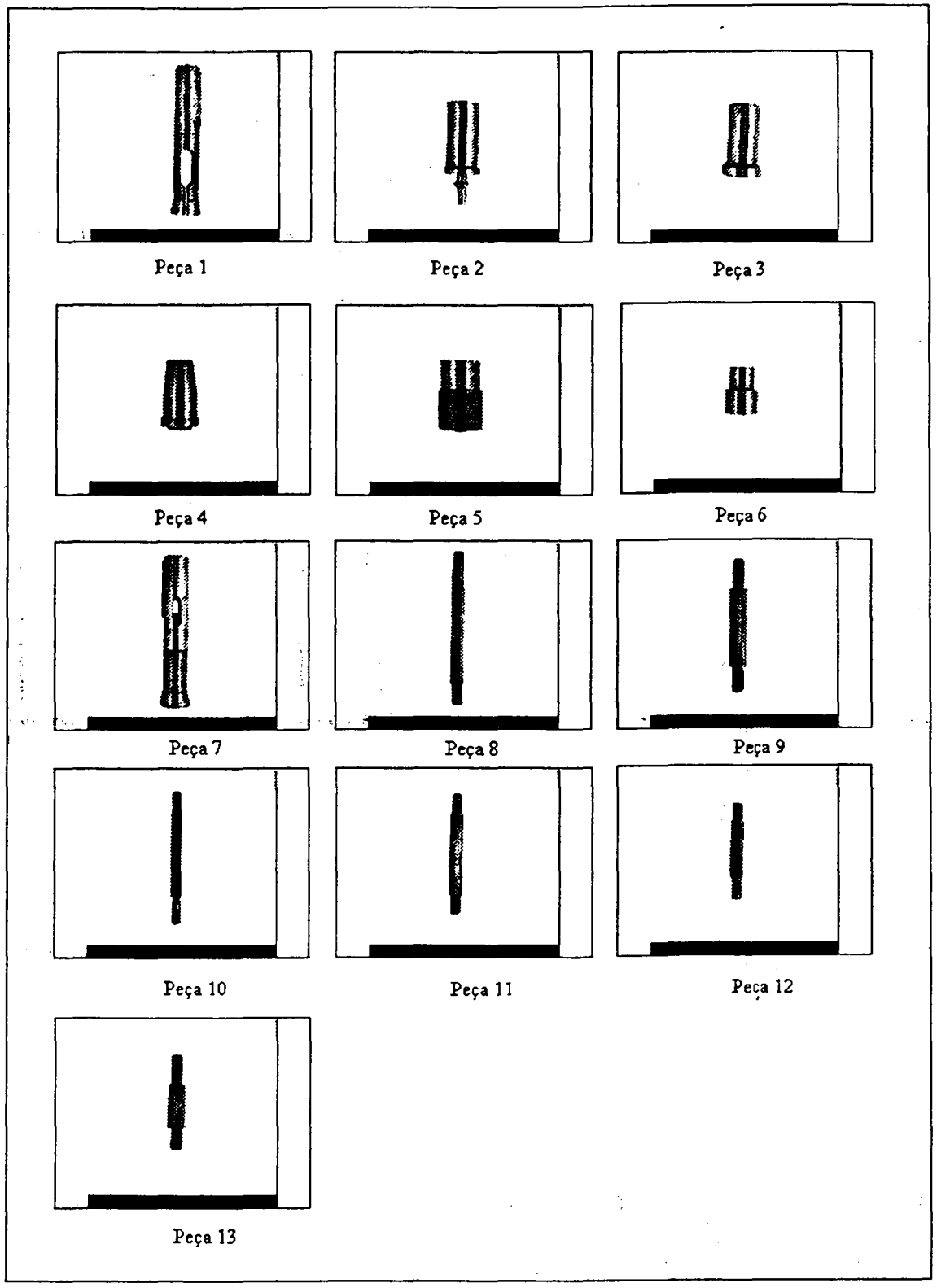

FIGURA 4.4 - Peças utilizandas para treinamento e reconhecimento

O formato do arquivo que foi utilizado para treinamento e reconhecimento de padrões e ângulos é mostrado na figura 4.5.a. e um exemplo deste arquivo é mostrado na figura 4.5.b.

<N° de neurônio na entrada> <N° de neurônios Kohonen> <N° de neurônios Grossberg>

<Erro de parada> <Alfa> <Beta>

<pontos do padrão 1 para entrada da rede neural>

<saída desejada do padrão 1>

<pontos do padrão 2 para entrada da rede neural>

<saída desejada do padrão 2>

.....

.....

<pontos do padrão **n** para entrada da rede neural>

<saída desejada do padrão **n**>

FIGURA 4.5.a- Modelo do arquivo de dados usado para treinar a rede neural

```
40 234 13
0.05 0.7 0.07
!CILR0.IMG
0.0000000000000 0.0000000000000 0.2790505886078 0.2642408609390 0.2283966392279 0.2047166228294
0.1876763701439 0.1940308064222 0.2775203287601 0.2524138092995 0.2617768943310 0.2291228622198
0.2061171978712 0.1902700066566 0.1917483955622 0.3194077312946 0.2828891873360 0.2263476550579
0.2736298441887 0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000
0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000 0.5248291492462
0.6417621374130 0.5591920614243 0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000
0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000
1 0 0 0 0 0 0 0 0 0 0 0 0
!CILR20.IMG
0.0000000000000 0.0000000000000 0.2300712913275 0.2385235428810 0.2098975330591 0.2139641791582
0.2036779522896 0.2186687290668 0.2720933258533 0.2862069904804 0.2699138224125 0.2330747693777
0.2041032165289 0.1992391943932 0.2003289461136 0.3366546928883 0.3234713077545 0.2565709650517
0.1504128277302 0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000
0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000 0.2194794416428 0.4422850012779
0.6232849359512 0.5508199334145 0.2310838997364 0.1045287996531 0.0000000000000 0.0000000000000
0.0000000000000 0.0000000000000 0.0000000000000 0.0000000000000
1 0 0 0 0 0 0 0 0 0 0 0 0
```

FIGURA 4.5.b - Exemplo do arquivo de pontos gerado pelo módulo CURVA

O fluxograma do módulo CURVA é mostrado na figura 4.6.

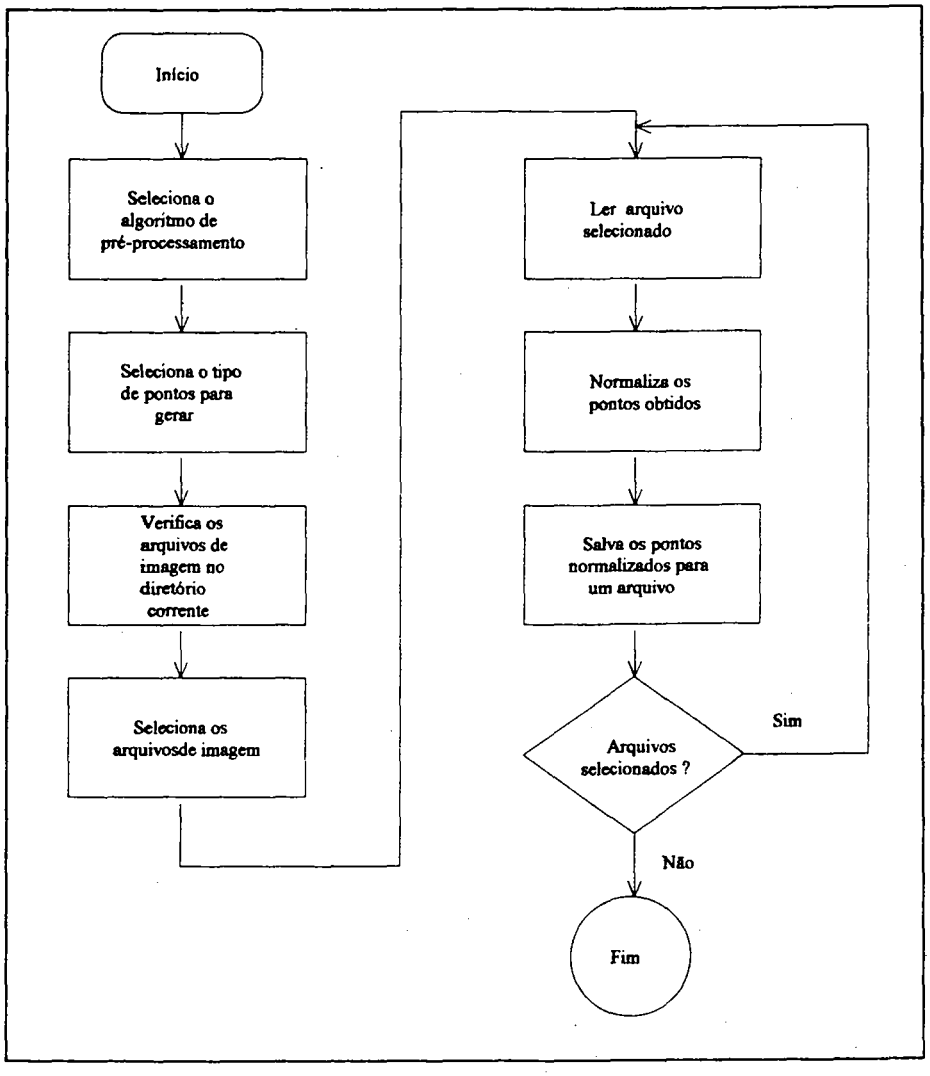

FIGURA 4.6 - Fluxograma do módulo CURVA

A figura 4.7 apresenta a tela de entrada do módulo CURVA

PROGRAMA PARA GERAR E PLOTAR OS PONTOS DA REDE NEURAL

(1) Gerar curvas de distribuição de intensidade

(2) Gerar o histograma

(3) Sai do programa

Escolha uma opção:

FIGURA 4.7 - Tela de apresentação do módulo CURVA

Na primeira tela existem 2 opções:

- Selecionando a opção (1) pode-se gerar os pontos para rede neural utilizando as curvas de intensidade.
- Selecionando a opção (2) obtém-se os pontos para entrada da rede através do histograma de níveis de cinza.

Selecionando-se ou a opção (1)ou a opção (2) tem-se 4 escolhas possíveis como mostrado na figura 4.8.

CURVAS DE DISTRIBUIÇÃO DE INTENSIDADE

(1) Gera somente os pontos da rede para treinamento de padrões

(2) Gera somente os pontos da rede para treinamento do ângulo de rotação

(3) Plota as curvas de distribuição de intensidade

(4) Volta ao menu anterior

Escolha uma opção:

FIGURA 4.8 - Tela que mostra as opções para gerar o arquivo de pontos (as mesmas opções são disponíveis para o histograma de níveis de cinza).

- Na opção (1) pode-se gerar somente os pontos para rede, sem entretanto mostrar as curvas de distribuição de intensidade ou o histograma, na tela.
- Com a opção (2) são gerados os pontos para treinar a rede neural que fará o reconhecimento do ângulo de rotação de cada peça.

- Na opção (3) pode-se plotar na tela as curvas de distribuição de intensidade de cada imagem selecionada ou o histograma de cada imagem.

A próxima tela exibida mostra os arquivos de imagem contidos no diretório corrente. Isto pode ser visto na figura 4.9.

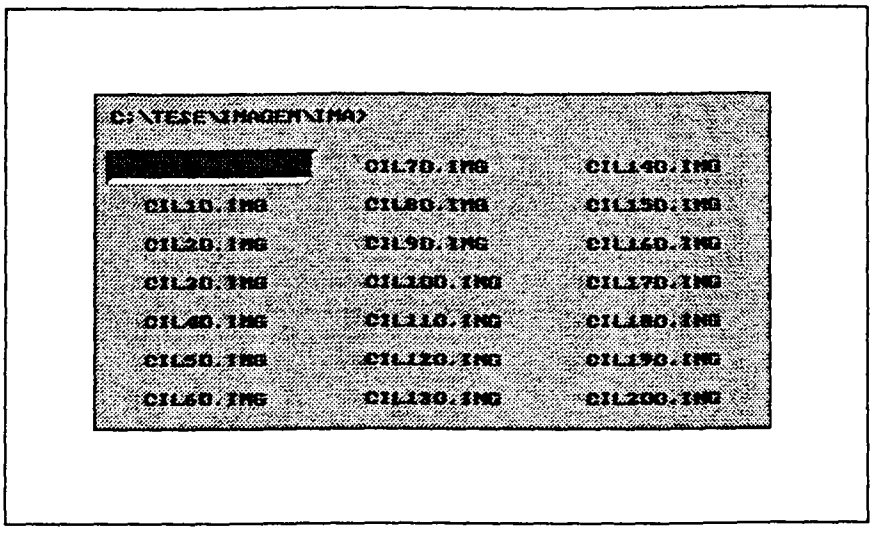

FIGURA 4.9 - Tela que mostra os arquivos de imagens que serão solucionados.

Na figura 4.9 pode-se ver os arquivos de imagem para serem pré-processados. A seleção pode ser feita pressionando-se [ENTER] e a movimentação de retângulo que indica a imagem que pode ser selecionada é feita com as teclas [←], [↑], [→] e [↓]. O pressionamento de [ENTER] sobre uma imagem já selecionada faz com que ela deixe de esta selecionada. Para que o software prossiga basta pressionar [SPACEBAR].

De posse do arquivo contendo os pontos para a entrada da rede que representam as imagens, pode-se agora iniciar o processo de treinamento da rede neural.

4.3.2 - RPNEURAL - Software para Reconhecimento de Peças usando Redes Neurais

Com este módulo pode-se treinar a rede neural utilizando o arquivo de pontos gerado pelo módulo CURVA bem como realizar a identificação "*on line*" das peças já conhecidas. O

treinamento poderá ser realizado de uma só vez ou em várias vezes. Neste último caso, pode-se salvar os pesos, ou seja, a base de conhecimento e no momento da retomada do treinamento esta base de conhecimento deve ser carregada para que o treinamento prossiga do ponto onde parou. Neste caso, são salvos também todos os parâmentros que influenciam no processo de aprendizado, de outro estaria-se interferindo no processo de aprendizado, sem contudo saber quais efeitos poderiam surgir desta interferência.

O fluxograma do módulo RPNEURAL é mostrado na figura 4.10.

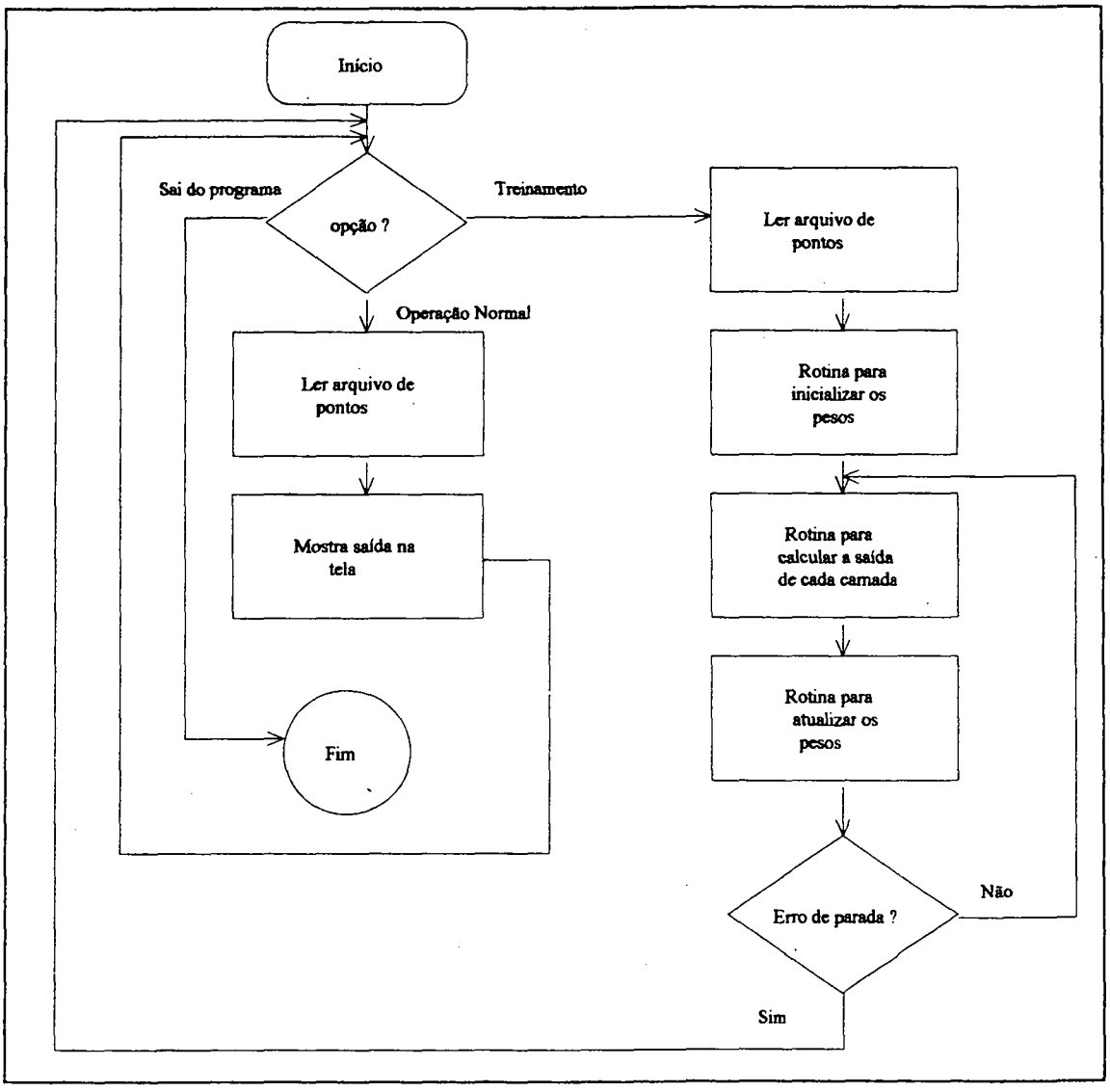

FIGURA 4.10 - Fluxograma do módulo RPNEURAL.

Um exemplo da saída produzida por módulo pode ser vista na figura 4.11. A letra **D**, antes do padrão significa que este é o padrão desejado, o padrão obtido é mostrado logo abaixo do desejado.

```
D1->1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 1->0.98 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
D2->1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 2->0.98 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
D3->1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 3->0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.98 0.00 0.00 0.00 0.00 0.00
D4->1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 4->0.98 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
D5->1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 5->0.98 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
D6->1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 6->0.98 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
D7->1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 7->0.98 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

FIGURA 4.11 - Exemplo da saída produzida pelo módulo RPNEURAL

No exemplo da figura 4.11, estão contidas as saídas desejadas e as saídas obtidas de uma mesma peça (Peça 1), cada uma delas corresponde a uma determinada imagem contida ou não no conjunto de treinamento usado. Pode-se notar, por exemplo, que a imagem 3 (D3) não foi reconhecida. O conjunto de padrões mostrados no exemplo não participou do treinamento da rede neural, mostrando com isso que a rede reconhece padrões que não foram apresentados para ela durante o treinamento.

Infelizmente no caso de um erro de reconhecimento a rede irá informar uma peça errada (por exemplo, o padrão D3 da figura 4.11). Possíveis soluções para melhorar esta taxa de acerto são comentadas no capítulo 5 que trata das conclusões e das perspectivas futuras.

4.4 - Resultados Alcançados

4.4.1 - Treinamento para reconhecer as peças

Após grande número de tentativas experimentais a rede neural foi implementada com 40 neurônios na camada de entrada, 234 neurônios na camada Kohonen e 13 neurônios na camada Grossberg. Foram utilizadas 18 imagens de cada peça tomadas de 20° em 20°. Esta rede foi treinada para reconhecer as peças independente de sua translação e rotação. Foram necessárias 12.000 iterações em aproximadamente 35 minutos num IBM PC-386DX40, para um erro de parada de 0.05, com parâmetros alfa=0.7 e beta=0.07. Com esta rede foi obtida uma taxa de acerto de cerca de 85.4% (68 erros em 468 imagens). A curva do erro é mostrada na figura 4.12.

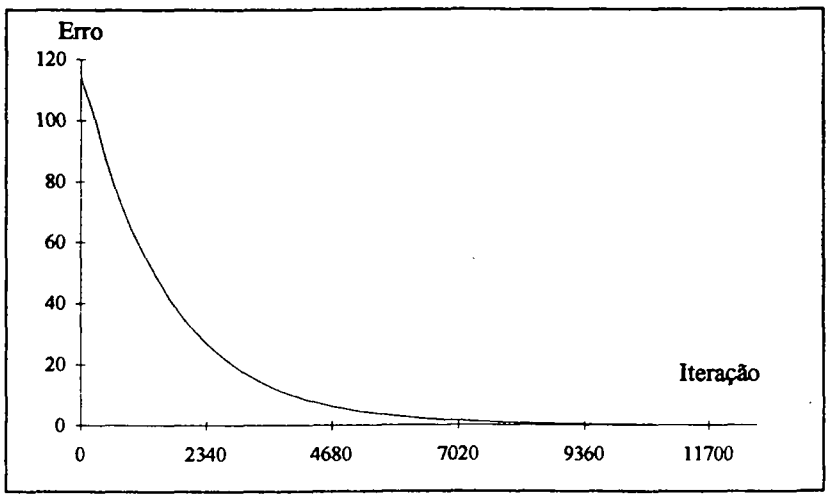

FIGURA 4.12 - Curva do erro do treinamento da rede para reconhecer as peças utilizando curvas de distribuição de intensidade.

Vários números de neurônios na entrada, na camada Kohonen e na Camada Grossberg foram utilizados.Por exemplo, foram utilizadas redes com 80 e com 120 neurônios na camada de entrada; os resultados obtidos foram praticamente os mesmos, contudo o tempo de treinamento cresceu bastante, como já era de se esperar. Os valores para os parâmetros Alfa e Beta também foram obtidos após vários testes durante o treinamento. Não se pode entretanto, garantir que estes seriam os melhores valores, isso porque teria-se que ser exaustivo durante os

testes ou provar matematicamente por algum teorema a veracidade da afirmação. Infelizmente nenhuma das duas opções é possível de ser verificada.

O número de neurônios na camada Kohonen foi determinado levando-se em conta que o número de neurônios nela esta intimamente relacionado com o número de padrões de treinamento sendo utilizados. A distribuição dos pesos desta camada deve ser tal que padrões bastantes similares devem ativar o mesmo neurônio e padrões diferentes devem ativar neurônios diferentes. Por isso, utilizou-se o número de padrões como sendo o número de neurônios nesta camada. Um número maior de neurônios poderia ter sido utilizado, mas eles provavelmente não iriam representar nenhum padrão e só dificultariam o treinamento. Já no caso de utilizar-se um número menor de neurônios, a precisão da rede poderia ser menor; alguns neurônios teriam que representar mais de um padrão de treinamento.

A escolha do número de neurônio na camada Grossberg foi mais simples. Cada neurônio representa uma determinada peça, das 13 utilizadas para treinamento e testes.

A tabela 4.1 mostra o comportamento da rede neural no processo de reconhecimento de peças utilizando curvas de distribuição de intensidade. A cada peça correspondem 18 padrões. Diz-se que saída desejada é ***n*** quando um determinado padrão de entrada deve ativar o neurônio nomeado com ***n***. Por exemplo, um padrão com saída desejada **1 0 0 0 0 0 0 0 0 0 0 0** terá que ativar o neurônio 1, outro padrão com saída desejada **0 0 0 1 0 0 0 0 0 0 0 0** terá que ativar o neurônio 4. A tabela 4.1 mostra o comportamento da rede neural usada para reconhecer as peças (nela estão contidos somente os padrões que não foram reconhecidos). Tomando como exemplo a primeira linha desta tabela, pode-se ver que os padrões 3 e 9 que tinham como saída desejada **1 0 0 0 0 0 0 0 0 0 0 0** e foram obtidas **0 0 0 0 0 0 1 0 0 0 0 0** e **0 0 0 0 0 0 1 0 0 0 0 0**, respectivamente.

Comportamento da saída da rede no processo de reconhecimento de peças utilizando curvas de distribuição de intensidade.		
Padrões de entrada errados	Saída desejada	Saída obtida
3 e 9	1	8 e 7
22, 24, 26, 28 e 36	2	13, 12, 13, 3 e 3
38 e 39	3	4 e 4
57	4	5
113, 114, 116, 121, 122 e 123	7	8, 1, 1, 1, 8 e 1
127 a 130, 132 a 139 e de 141 a 144	8	7 para 133 e 142 1 para o restante
148, 149, 150, 151, 157, 158, 159 e 160	9	1, 11 e 10 para o resto
163 a 168, 170, 171, 172, 173, 179 e 180	10	9, 11, 9, 11, 11, 9, 9, 11, 11, 11, 9 e 9
186, 192, 195, 196	11	12, 10, 10 e 10
201, 205 e 210	12	2, 2 e 2
218, 219, 220, 221, 223, 227, 228, 229, 230, 231 e 232	13	12 para 221, 229, 230 e 231 2 para o restante

Tabela 4.1 - Comportamento da rede neural no reconhecimento das peças.

A mesma rede foi treinada utilizando agora pontos obtidos com a utilização de histograma de níveis de cinza. Foram necessárias aproximadamente 12.600 iterações e cerca 40 minutos usando o mesmo equipamento (os parâmetros usados são os mesmos do caso anterior). A taxa de acerto foi de 87% (59 erros em 468 imagens). A curva de erro para esta rede é mostrada na figura 4.13.

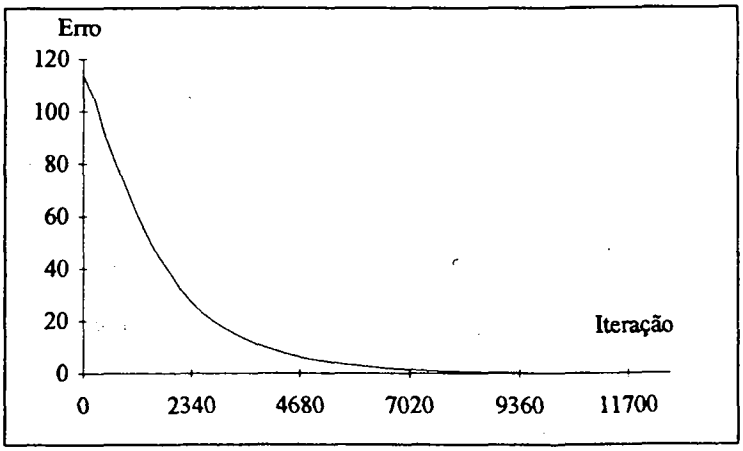

FIGURA 4.13 - Comportamento do erro durante o treinamento da rede utilizando pontos gerados com histograma.

A tabela 4.2 mostra os resultados obtidos com a utilização do histograma como algoritmo de pré-processamento (padrões não reconhecidos).

Comportamento da saída da rede no processo de reconhecimento de peças utilizando histograma de níveis de cinza.		
Padrão	Saída desejada	Saída obtida
15 e 17	1	4 e 7
20, 21, 24, 26 e 35	2	3, 7, 7, 3, 3
37, 39, 42, 43, 46, 47, 48 e 51	3	1, 2, 1, 2, 1, 7, 2 e 1
57	4	7
-	5	-
91, 93, 94, 95, 96, 97, 101 e 106	6	2, 1, 4, 1, 4, 4, 1 e 4
111, 115, 119, 123 e 125	7	1, 2, 1, 1 e 1
129, 132, 133, 136, 139 e 141	8	5, 9, 9, 4, 13 e 11
147, 152 e156	9	8, 8 e 11
164,169, 170, 174, 177, 178, 185, 187 e 195	10	12, 8, 8, 5, 5, 12, 9, 9 e 8
196	11	12
200, 202, 205, 208 e 210	12	11, 11, 10, 10 e 10
218, 219, 223, 225, 228 e 232	13	9, 11, 11, 4, 9 e 9

Tabela 4.2 - Comportamento da rede neural no reconhecimento das peças utilizando histograma de níveis de cinza.

4.4.2 - Treinamento da rede para reconhecer o ângulo de rotação da peça

Foi utilizada uma rede com 40 neurônios na camada de entrada, 156 neurônios na camada Kohonen e 2 neurônios na camada Grossberg. Como saída desejada eram fornecidos o cosseno e o seno do ângulo. No treinamento desta rede foram usadas imagens de 30° em 30° de cada peça e no modo de operação normal foi utilizada uma interpolação linear para auxiliar na precisão dos resultados. Foram necessárias 8.000 iterações em aproximadamente 15 minutos. As curvas mostrando o comportamento do erro são mostradas nas figura 4.14 (utilizando curvas de distribuição de intensidade) e figura 4.15 (utilizando histograma).

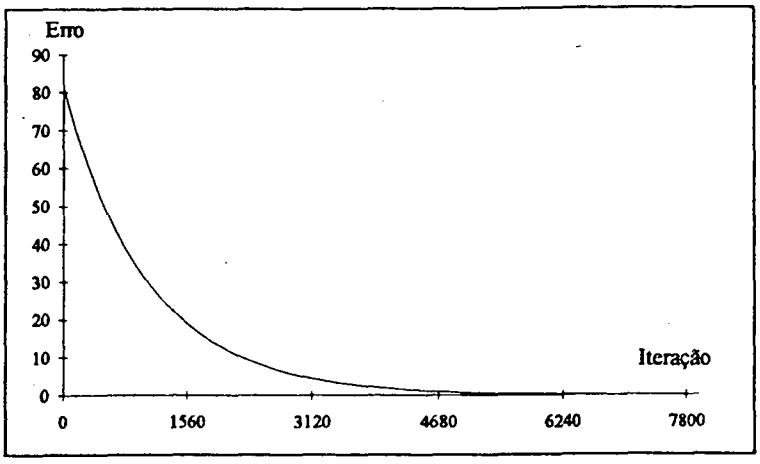

Figura 4.14 - Comportamento do erro com a utilização das curvas de distribuição de intensidade.

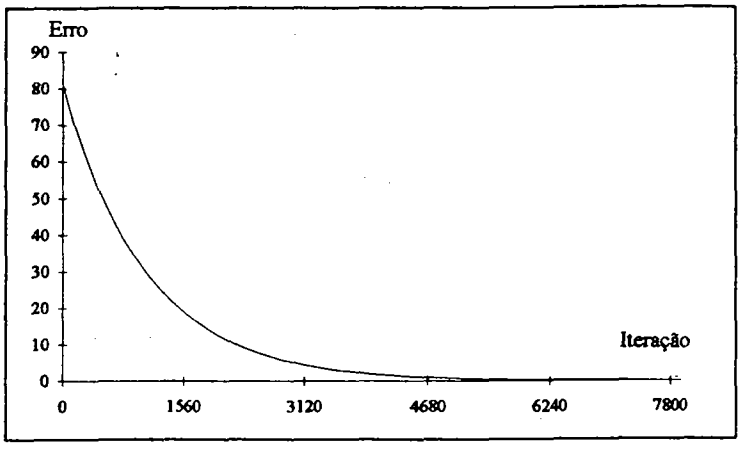

Figura 4.15 - Comportamento do erro com a utilização do histograma de níveis de cinza.

Durante o treinamento a rede era treinada de modo a existir somente um neurônio ganhador da competição. Já durante o reconhecimento os dois neurônios com maior valor de saída eram declarados ganhadores da competição. Entretanto, a somatória das saídas desses dois neurônios não pode ultrapassar 1 , porque se somente um neurônio fosse declarado ganhador sua saída seria levada a 1 e as saídas dos restantes seriam levadas a 0. Os valores de saída de cada neurônio obtidos no final do treinamento, são salvos junto com a base de conhecimento. Esses valores são aqueles obtidos com a aplicação dos padrões de treinamento, ou seja, se um neurônio foi treinado para reconhecer uma determinada peça rotacionada de 30°, quando da aplicação deste padrão na entrada da rede, esse neurônio terá seu maior valor de saída. Esses valores são utilizados, para dizer o quão próximo deste valor a saída de um determinado padrão esta.

Para que se pudesse associar a cada neurônio da camada Kohonen um determinado ângulo, o treinamento desta camada deixou de ser não supervisionado e passou a ser determinístico, ou seja, padrões rotacionados de um determinado ângulo, ativam um único neurônio pré-especificado.

Com esta estrutura de rede conseguiu-se uma taxa de acerto de aproximadamente 74%, para os tipos de pré-processamento aqui utilizados, com uma precisão de ± 10°.

Para tentar melhorar a taxa de acerto da rede neural utilizada para reconhecer a rotação angular da peça foi treinada uma nova rede neural tendo 40 neurônios na camada de entrada, 234 neurônios na camada intermediária e 2 neurônios na camada de saída. Durante o treinamento foram utilizadas 18 imagens de cada peça tomadas de 20° em 20°. Foram usados os dois algoritmos de pré-processamento já descritos. Para rede treinada com pontos gerados com a curva de distribuição de intensidade, obteve-se uma taxa de acerto de 77,7%, e para a rede utilizando os pontos gerados com histograma de níveis de cinza obteve-se 81,6 % e precisão também de ± 10°. As duas rede utilizaram os seguintes parâmetros: alfa = 0.7, beta = 0.07 e erro de parada = 0.005. O tempo de treinamento nos dois casos foi de aproximadamente 30 minutos em cerca de 16.000 iterações. As curvas do erro médio quadrático para os dois casos são mostradas nas figuras 4.16 e 4.17.

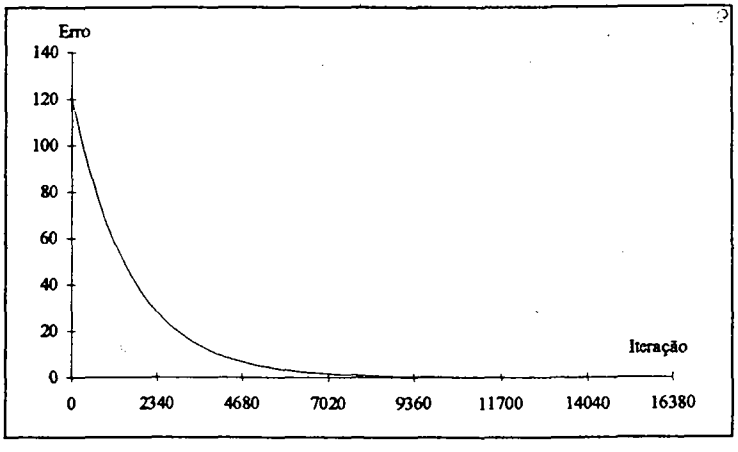

FIGURA 4.16 - Curva de erro da rede utilizando pontos gerados as curvas de distribuição de intensidade

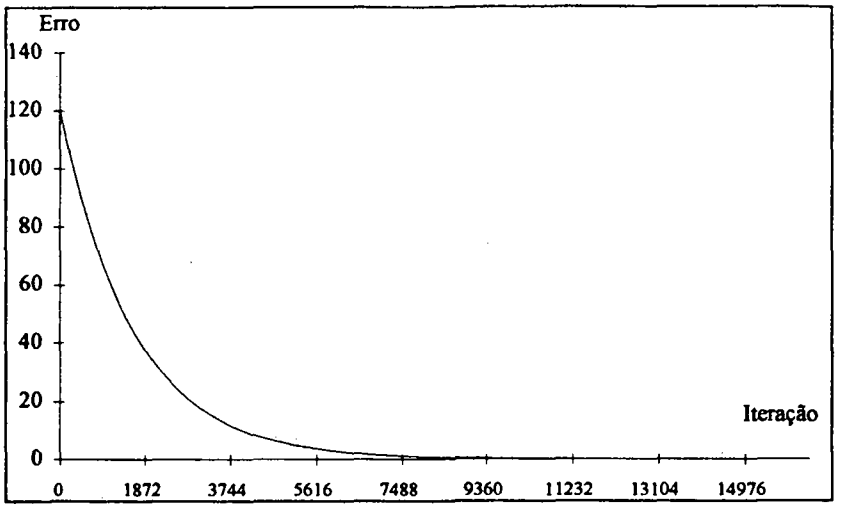

FIGURA 4.17 - Curva de erro da rede utilizando pontos gerados com histograma de níveis de cinza

4.5 - Análise dos Resultados

Os resultados obtidos com a utilização dos dois algoritmos de pré-processamento são praticamente iguais quando do reconhecimento dos padrões e quando usa-se a rede para reconhecimento do ângulo com apenas 156 neurônios na camada intermediária. Deste modo, considerando que os dois algoritmos são bastantes simples, um possível melhoramento da taxa de acerto pode ser obtido com algum outro algoritmo que melhor extraia as características da peça contidas na imagem. Consegue-se uma taxa de acerto um pouco melhor para a rede que reconhece o ângulo de rotação utilizando o histograma de níveis de cinza e 234 neurônios na camada intermediária. É importante salientar também, que se for conseguida uma iluminação que permita rotacionar e transladar a peça sem que os pixel que representam uma determinada área mudem os seus tons de cinza, os resultados obtidos serão melhores.

CAPÍTULO V

CONCLUSÕES E PERSPECTIVAS

Neste trabalho foi desenvolvido um sistema que permite a identificação de peças mecânicas rotativas. A imagem de cada peça é obtida por uma câmera CCD e digitalizada. A imagem digitalizada é processada de forma a permitir uma identificação da peça após uma etapa inicial de aprendizado neural. Além disso, o sistema desenvolvido é capaz de fornecer a posição e orientação da peça.

O sistema aqui desenvolvido e implementado, possui uma taxa de acerto considerada aceitável, tendo em vista que outros sistemas de reconhecimento possuem taxas de acerto entre 72% e 100% [9]. Tem-se que ressaltar a dificuldade para fazer este tipo de comparação, a qual foi explicada no capítulo 3. Existem algumas razões para que esta taxa de acerto não seja maior. Foi utilizada uma iluminação não simétrica, ou seja, usou-se duas lâmpadas fluorescentes de cada lado da estrutura utilizada para obter as imagens. Como os dois algoritmos de pré-processamento utilizados são bastantes simples e baseiam-se no nível de cinza de cada pixel da imagem, esta deficiência na iluminação tende a mudar os níveis de cinza de alguns pixels que representam uma determinada área da peça sendo vista, dependendo da rotação e translação da mesma. Uma possível melhora pode vir a ser obtida fazendo-se a iluminação usando *spots* com lâmpadas especiais (conhecidas como "Luz dia"). Esse tipo de lâmpada é incandescente, mas emite luz branca.

Outro motivo é a semelhança de algumas peças usadas no processo de treinamento e reconhecimento. Algumas delas diferem basicamente no tamanho (sensibilidade de escala), sendo que este tipo de similaridade mostrou-se difícil de ser totalmente "compreendido" ou separado pela rede neural. Além disso, peças sobrepostas não podem ser reconhecidas devido à simplicidade dos algoritmos utilizados para pré-processar as imagens.

Uma possível solução seria a utilização de redes com inibição lateral, que podem representar padrões mais complexos. Um exemplo destas redes seria as chamadas EXIN [14] (*excitatory+inhibitory*). Redes EXIN auto-organizam-se em complexos ambientes perceptuais, na presença de múltiplos padrões sobrepostos, múltiplas escalas e incertezas. A rede usa uma nova regra de aprendizado inibitório, em adição a uma regra de aprendizado excitatório, para permitir a superposição de múltiplas ativações neurais (múltiplos ganhadores), ao invés de um só ganhador.

Algoritmos de pré-processamento que permitam melhor extrair as características das peças também podem melhorar a taxa de acerto da rede neural.

Com a disponibilidade da Célula Flexível de Montagem miniaturizada adquirida pelo LCMI, outras técnicas de iluminação podem ser experimentadas. O capítulo 3 ilustra algumas destas técnicas.

A utilização de redes neurais para reconhecer padrões poderá ser uma das saídas para sistemas de visão por computador, como demonstrado neste trabalho. Futuros trabalhos utilizando novas estruturas de redes neurais ou variações de uma já conhecida e ainda a experiência relatada aqui, poderam resultar em produtos robustos e de boa precisão que poderiam facilmente serem aplicados na indústria.

Outra perspectiva de interesse seria o estudo de técnicas para o reconhecimento de peças em 3D. Para tal, seria necessário aquirir a imagem da peça com 2 câmaras CCD. Uma possível aplicação disto na Célula Flexível de Manufatura aqui descrita seria a identificação de peças prismáticas

REFERÊNCIAS BIBLIOGRÁFICAS

[1] - Wasserman, Philip D.: *Neural computing: theory and practice*, Van Nostrand Reinhold,. 1989.

[2] - Freeman, James A. e Skapura, David M.: *Neural networks: algoritms, aplications, and programming techniques*, Addison-Wesley Publishing Company, Inc., 1991.

[3] - Doyhoff, Judith.: *Neural network architectures: an introdution*, Van Nostrand Reinhold, 1990.

[4] - Lawrence, Jeannette J.: *Untangling neural nets*, Dr. Dobb's Journal, Abril 1990.

[5] - King, Todd.: *Using neural networks for pattern recognition*, Dr. Dobb's Journal Janeiro 1989.

[6] - Blum, Adam: *Bidirectional Memory Systems in C++*, Dr. Dobb's Journal, Abril 1990.

[7] - Jones, William P. e Hoskins, Josiah: *Back-propagation: a generalized delta learning rule*, Byte, Outubro 1987.

[8] - Treleaven, Philip; Pacheco, Marco e Vellasco, Marley: *VLSI Architecture for Neural Networks*, IEE MICRO, Dezembro 1989.

[9] - Huang, Chien-nan; Lim, Chin-Choon e Liu, Ming C..*Comparison of Image Processing Algorithms and Neural Networks in Machine Vision Inspection*, Proceedings of the 14th Annual Conference on Computer and Industrial Engineering, Novembro 1992.

[10]- Cantoni, V.; Levialdi, S e Musso, G.: *Image Analysis and Processing*, Proceedings of the Third International Conference of Image Analysis and Processing, Setembro-Outubro 1985.

[11]- Castleman, Kenneth R.:*Digital Image Processing*, Prentice-Hall, Inc. 1979.

[12]- Pratt, William K.: *Digital Image Processing*, John Wiley & Sons. Inc. 1991.

[13]- Sepeda Filho, Idmilson H. e Stemmer, Marcelo R.: *Um Sistema de Reconhecimento de Peças baseado em Redes Neurais*, Aprovado para publicação nos Anais do 10° Congresso Brasileiro de Automatica, a realizar-se em Setembro, 1994.

[14]- Marshall, Jonathan A.: *Adaptive Perceptual Pattern Recognition by Self_Organizing Neural Networks: Context, Uncertainty, Multiplicity, and Scale*, FTP archieve.cis.ohio-state.edu (/pub/neuroprose/marshall.context.ps.Z), Fevereiro, 1993.

[15]- Perfetto, Juan C.: *Reconocimiento de Imágenes Aplicando Redes Neurais*, Revista Telegrafica-Electronica, Setembro 1989.

[16]- Carpenter, Gail A. e Grossberg, Stephen : *The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network*, IEEE Computer, Março 1988.

[17]- Linsker, Ralph : *Self-Organization in a Percetual Network*, IEEE Computer, Março 1988.

[18]- Feldman, Jerome A.; Fanty, Mark A. e Goddard: *Computing with Structured Neural Networks*, IEEE Computer, Março 1988.

[19]- Rabelo, Luis C. e Avula, Xavier J. R.: *Hierarchical Neuralcontroller Architecture for Robotic Manipulation*, IEEE Control Systems, Abril 1992.

[20]- Klimasauskas, Casey : *Neural Nets and Noise Filtering*, Dr. Dobb's Journal, Janeiro 1989.

[21]- Josin, Gary : *Neural-Network Heuristics*, Byte, Outubro 1987.

[22]- Czuchry, Andrew J. : *A Neural Network Instantiation Environment*, Dr. Dobb's Journal, Abril 1990.

[23]- Widrow, Bernard e Winter, Rodner : *Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition*, IEEE Computer, Março 1988.

[24]- Fukuda, Toshio : *Theory and Applications of Neural Networks for Industrial Control Systems*, IEEE Transactions on Industrial Electronics, Vol 39 N° 6, Dezembro 1992.

[25]- Farley, James F. e Varhol, Peter D. : *Neural Nets for Predicting Behavior*, Dr. Dobb's Journal, Fevereiro 1993.

[26]- Bärmann, Frank e Biegler-König, Friedrich : *On a Class of Efficient Learning Algorithms for Neural Networks*, Neural Networks Vol 5, 1992.

[27]- Klimasauskas, Casimir C. : *Neural Nets Tell Why*, Dr. Dobb's Journal, Abril 1991.

[28]- Poggio, T. e Edelman : *A network that learns to recognize three-dimensional objects*, Nature Vol 343, Janeiro 1990.

[29]- Barreto, Jorge Muniz e Azevedo, Fernando Mendes de : *Neural Networks: Theoretical Foundations and Applications*, draft de um livro que terá o mesmo título, Novembro 1993.

[30] - Hecht-Nielsen, Robert : *Neurocomputing: picking the human brain*, IEEE Spectrum, Março 1988.

[31] - Hopfield, John J. e Tang, David W. : *Computing with Neural Circuits: A Model*, Science, vol. 233, Agosto 1986.

[32] - Hecht-Nielsen, Robert: *Theory of the Backpropagation Neural Network*, IJCNN, Junho 1989.